Create Web Services with Java

6104-565-09

Create Web Services With Java

Written By

Rony Keren

Internet Team

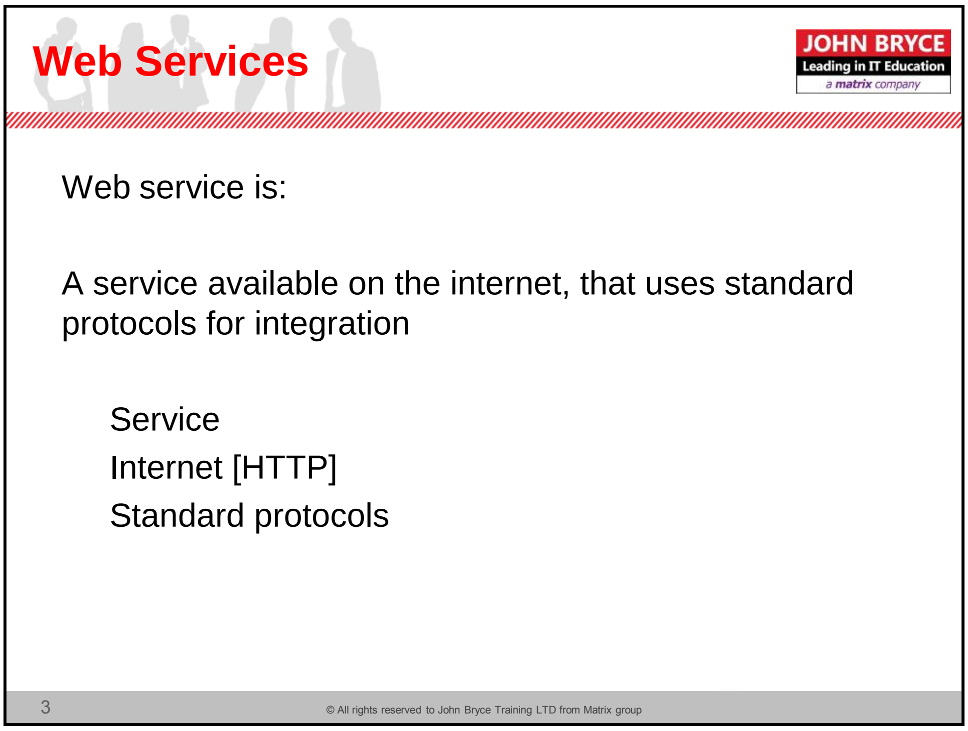
John-Bryce

JOHN BRYCE
Leading in IT Education
a matrix company

Topics

JOHN BRYCE
Leading in IT Education
a matrix company

- Definition
- Architecture
- RPC
- XML based Web Services
- REST based Web Services



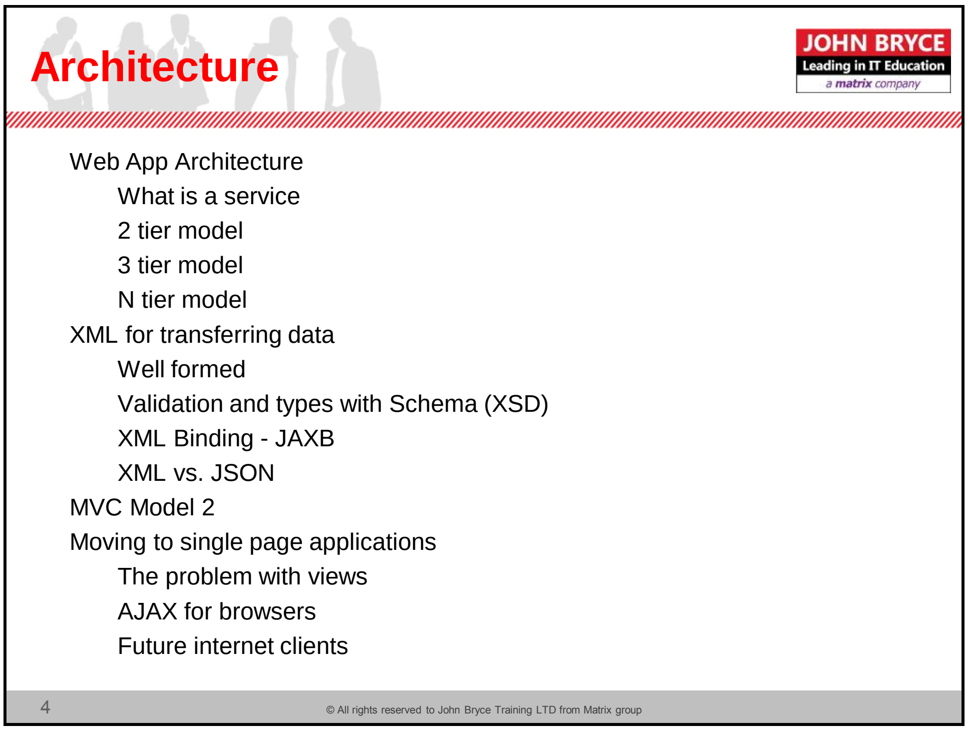
Web Services

Web service is:

A service available on the internet, that uses standard protocols for integration

- Service
- Internet [HTTP]
- Standard protocols

3



Architecture

- Web App Architecture
  - What is a service
  - 2 tier model
  - 3 tier model
  - N tier model
- XML for transferring data
  - Well formed
  - Validation and types with Schema (XSD)
  - XML Binding - JAXB
  - XML vs. JSON
- MVC Model 2
- Moving to single page applications
  - The problem with views
  - AJAX for browsers
  - Future internet clients

4

Architecture

JOHN BRYCE
Leading in IT Education
a matrix company

What is a service?

A model, provided by vendor, that allows clients to communicate and interact

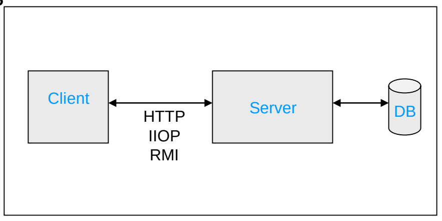
May be self-descriptive since a contract is needed

Architecture

JOHN BRYCE
Leading in IT Education
a matrix company

2 Tier Model

Tier 1 – container
Tier 2 – DB or any other 3rd party
Containers are focusing on communication
- Web containers – HTTP→ CGI
- RMI containers – Java connectors
- IIOP containers – IDL connectors



Disadvantages
- No infrastructure services
- Problematic when moving to large scales

Architecture

JOHN BRYCE Leading in IT Education a matrix company

3 Tier Model

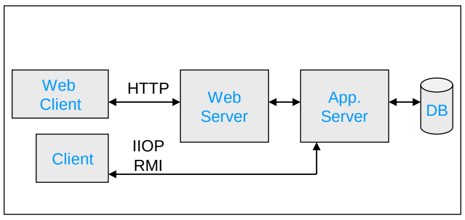
Tier 1 – Web server

Tier 2 – Business server

Tier 3 – DB or any other 3rd party

Business server provides infrastructural services

- Development focuses on service implementation
- Highly scalable
- Support various protocols



Architecture

JOHN BRYCE Leading in IT Education a matrix company

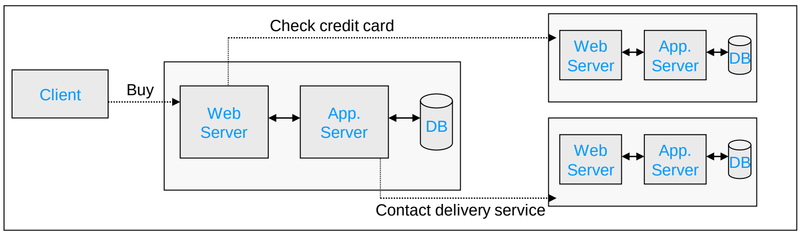
N Tier Model

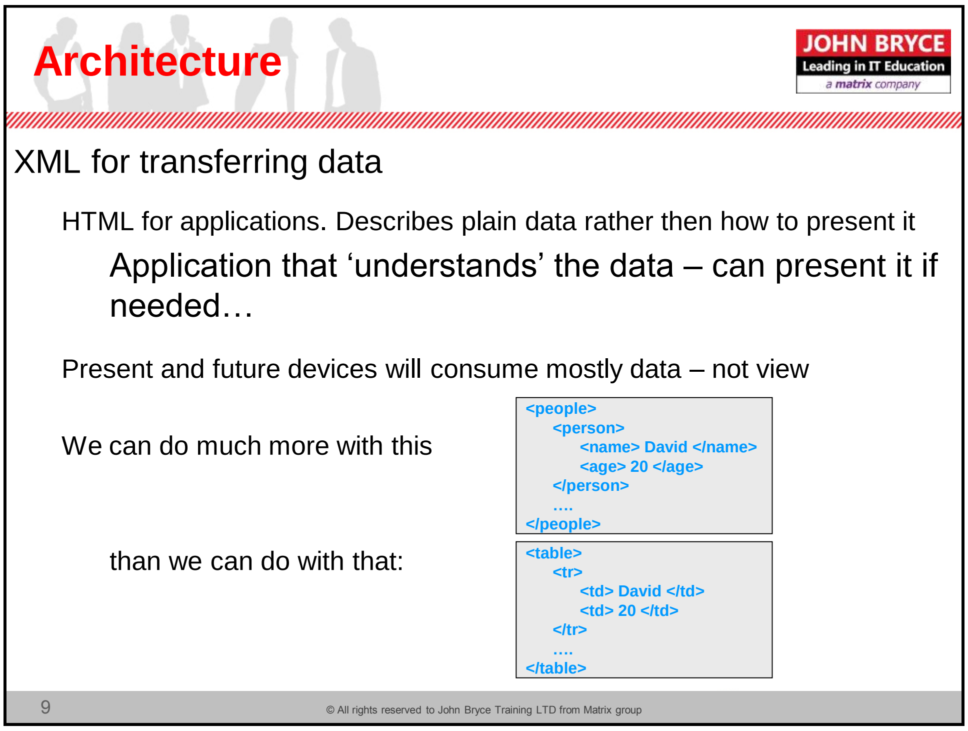
2, 3 tier systems interaction

Client request might be handled by multiple systems

One system must effectively interact with another

B2B / EAI / SOA …





Architecture

JOHN BRYCE
Leading in IT Education
a matrix company

XML for transferring data

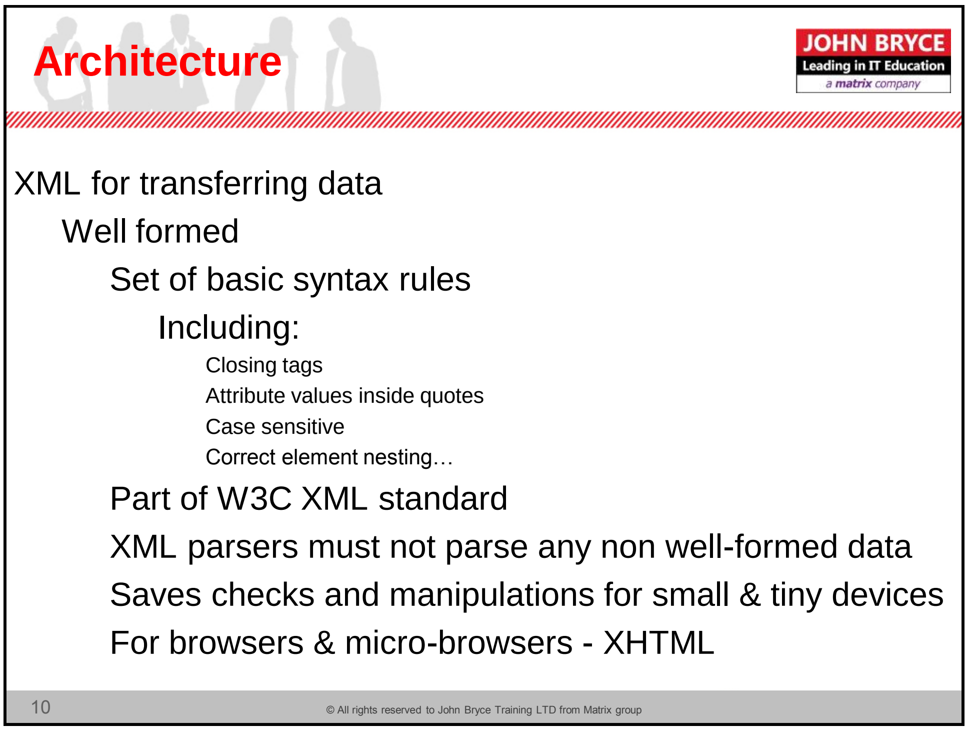
- HTML for applications. Describes plain data rather then how to present it
  - Application that ‘understands’ the data – can present it if needed…
- Present and future devices will consume mostly data – not view
- We can do much more with this

```
<people>
    <person>
        <name> David </name>
        <age> 20 </age>
    </person>
    ….
</people>
```

  - than we can do with that:

```
<table>
    <tr>
        <td> David </td>
        <td> 20 </td>
    </tr>
    ….
</table>
```

9 © All rights reserved to John Bryce Training LTD from Matrix group



Architecture

JOHN BRYCE
Leading in IT Education
a matrix company

XML for transferring data

- Well formed
  - Set of basic syntax rules
    - Including:
      - Closing tags
      - Attribute values inside quotes
      - Case sensitive
      - Correct element nesting…
  - Part of W3C XML standard
  - XML parsers must not parse any non well-formed data
  - Saves checks and manipulations for small & tiny devices
  - For browsers & micro-browsers - XHTML

10 © All rights reserved to John Bryce Training LTD from Matrix group

Architecture

JOHN BRYCE
Leading in IT Education
a matrix company

- XML for transferring data
  - Validation and types
    - XML structure is described via XSD (Schema)
    - W3C standard
    - XSD Schema defines:
      - Element name & content
      - Attributes
      - Simple and complex types
    - Since XSD defines primitives (xsd:integer, xsd:date….) – objects can be described as well..

11

Architecture XML for transferring data

Schema example:

JOHN BRYCE
Leading in IT Education
a matrix company

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="People">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Person" type="PersonType" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
     </xsd:element>
     <xsd:complexType name="PersonType">
        <xsd:sequence>
              <xsd:element name="Name" type="xsd:string"/>
              <xsd:element name="Age" type="AgeType"/>
              <xsd:element name="BirthDate" type="xsd:date" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="gender" type="GenderType" use="required"/>
    </xsd:complexType>
    <xsd:simpleType name="AgeType">
        <xsd:restriction base="xsd:nonNegativeInteger">
              <xsd:minInclusive value="0"/>
              <xsd:maxInclusive value="120"/>
       </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="GenderType">
       <xsd:restriction base="xsd:string">
              <xsd:enumeration value="M"/>
              <xsd:enumeration value="F"/>
       </xsd:restriction>
     </xsd:simpleType>
</xsd:schema>
```

12

Architecture XML for transferring data

Schema example:

```
<?xml version="1.0"?>
<People xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="PeopleSchema.xsd">
        <Person gender="M">
                <Name>Bill</Name>
                <Age>35</Age>
                <BirthDate>1984-04-13</BirthDate>
        </Person>
        <Person gender="F">
                <Name>Dana</Name>
                <Age>47</Age>
                <BirthDate>1961-11-03</BirthDate>
        </Person>
        <Person gender="F">
                <Name>Amy</Name>
                <Age>23</Age>
                <BirthDate>1991-04-15</BirthDate>
        </Person>
        <Person gender="M">
                <Name>David</Name>
                <Age>13</Age>
                <BirthDate>2000-07-02</BirthDate>
        </Person>
</People>
```

Architecture

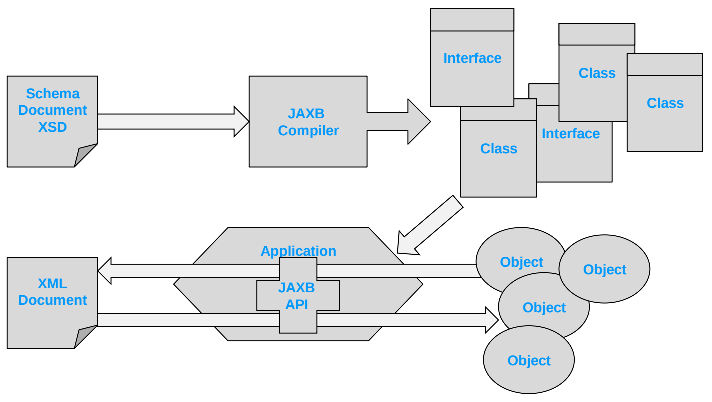
XML Binding
- Useful when developing application that integrates via XML
- Complex data structures are mapped to classes
  - Better approach then DOM

Java API for XML Binding – JAXB is included in JDK 7

Architecture

JOHN BRYCE Leading in IT Education a matrix company

JAXB

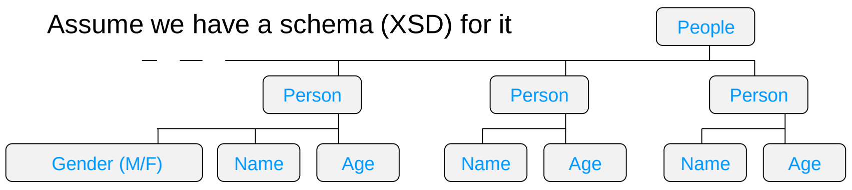


Architecture

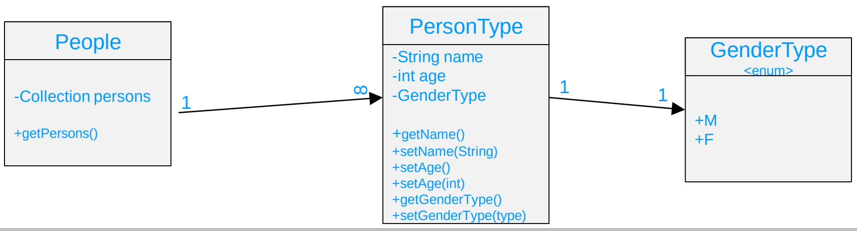
JOHN BRYCE Leading in IT Education a matrix company

JAXB

Assume we have a schema (XSD) for it



JAXB generated classes



Architecture

JOHN BRYCE
Leading in IT Education
a matrix company

With JAXB we get:

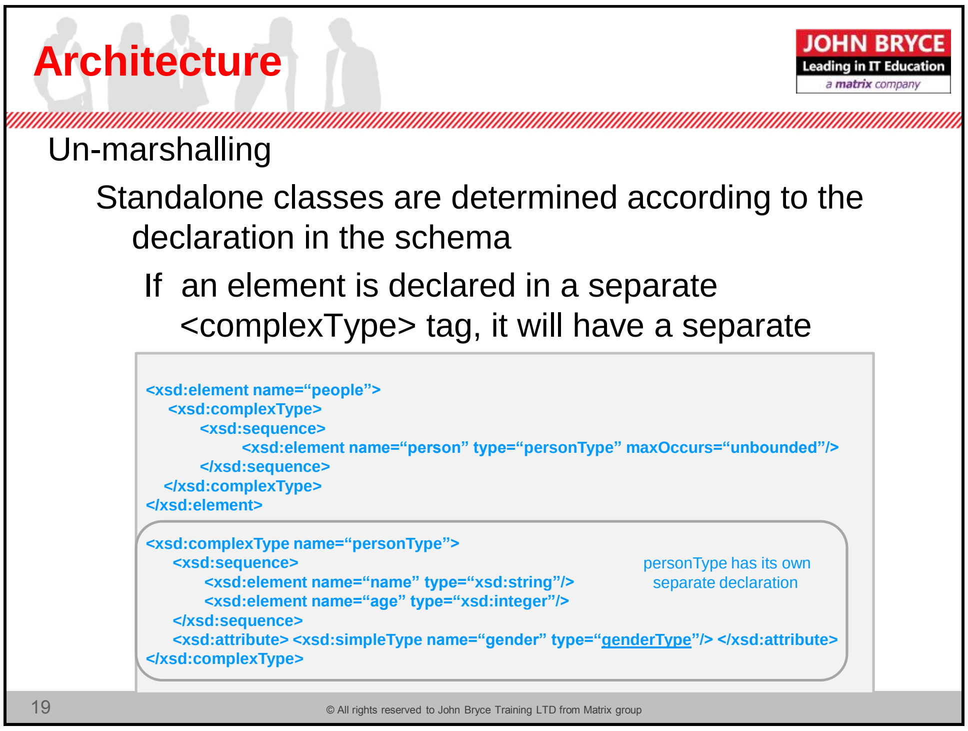
- Generated classes according to schema
- Lightweight Java objects
- Auto Marshalling and Un-marshalling

Architecture

JOHN BRYCE
Leading in IT Education
a matrix company

Un-marshalling

- Convert from XML to objects
- Classes are generated by Binding Compiler
  - Each complex type is turned into
    - Interface (or inner interface)
    - Class Implementation (or inner class)



Architecture

JOHN BRYCE
Leading in IT Education
a matrix company

Un-marshalling

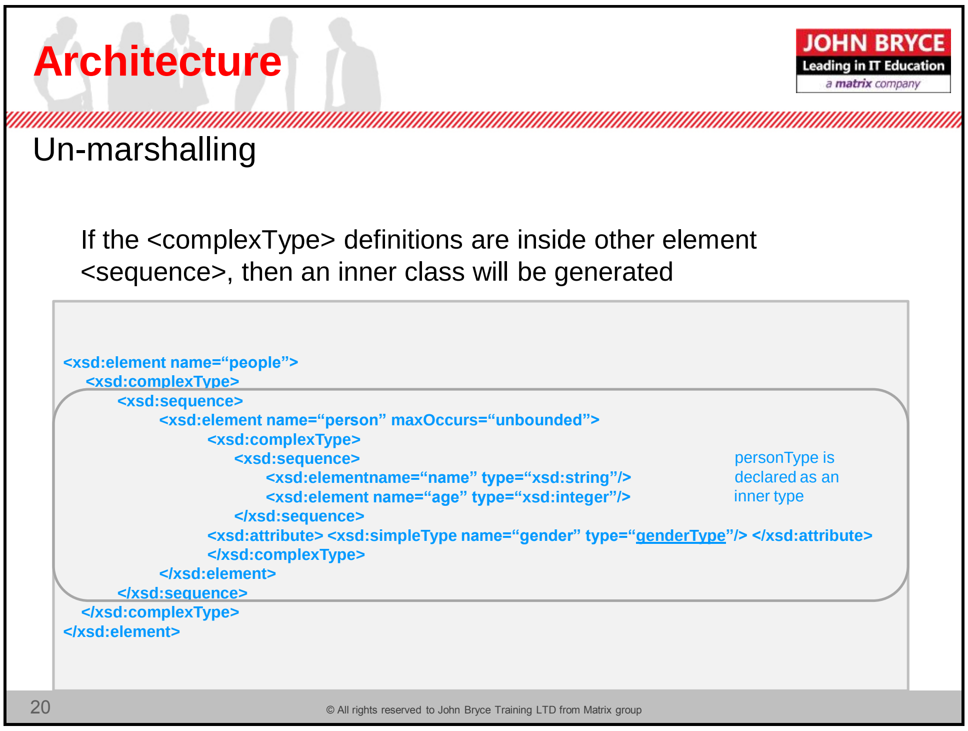
Standalone classes are determined according to the declaration in the schema

If an element is declared in a separate <complexType> tag, it will have a separate

```
<xsd:element name="people">
    <xsd:complexType>
         <xsd:sequence>
              <xsd:element name="person" type="personType" maxOccurs="unbounded"/>
         </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="personType">
     <xsd:sequence>
          <xsd:element name="name" type="xsd:string"/>
          <xsd:element name="age" type="xsd:integer"/>
     </xsd:sequence>
     <xsd:attribute> <xsd:simpleType name="gender" type="genderType"/> </xsd:attribute>
</xsd:complexType>
```

personType has its own separate declaration

19



Architecture

JOHN BRYCE
Leading in IT Education
a matrix company

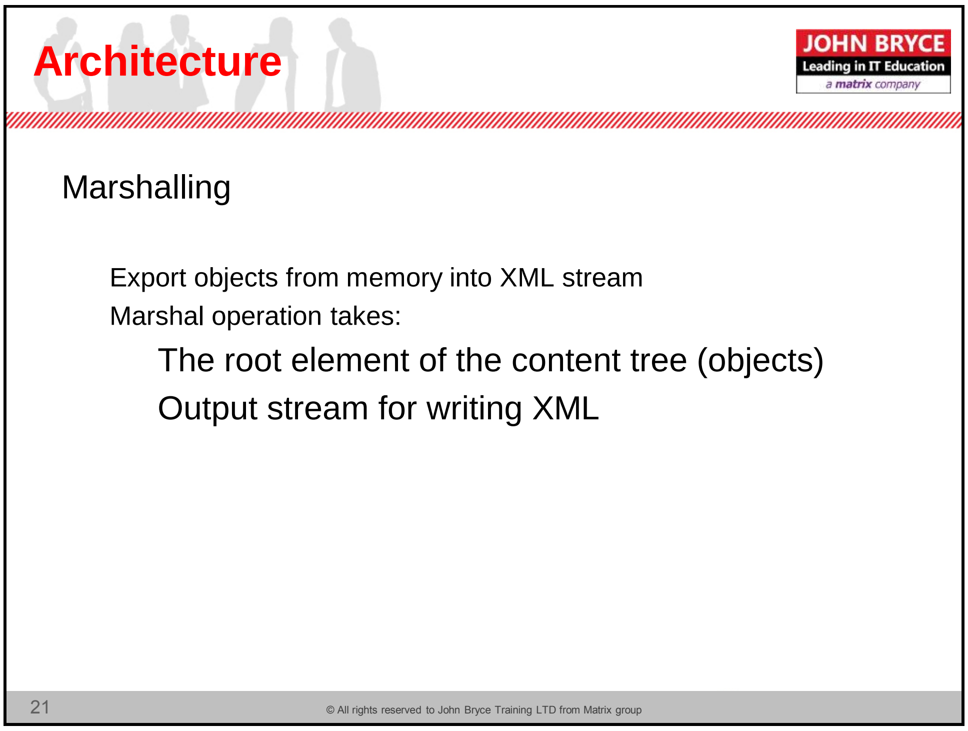
Un-marshalling

If the <complexType> definitions are inside other element <sequence>, then an inner class will be generated

```
<xsd:element name="people">
    <xsd:complexType>
         <xsd:sequence>
              <xsd:element name="person" maxOccurs="unbounded">
                   <xsd:complexType>
                        <xsd:sequence>
                             <xsd:elementname="name" type="xsd:string"/>
                             <xsd:element name="age" type="xsd:integer"/>
                        </xsd:sequence>
                   <xsd:attribute> <xsd:simpleType name="gender" type="genderType"/> </xsd:attribute>
                   </xsd:complexType>
              </xsd:element>
         </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

personType is declared as an inner type

20



Architecture

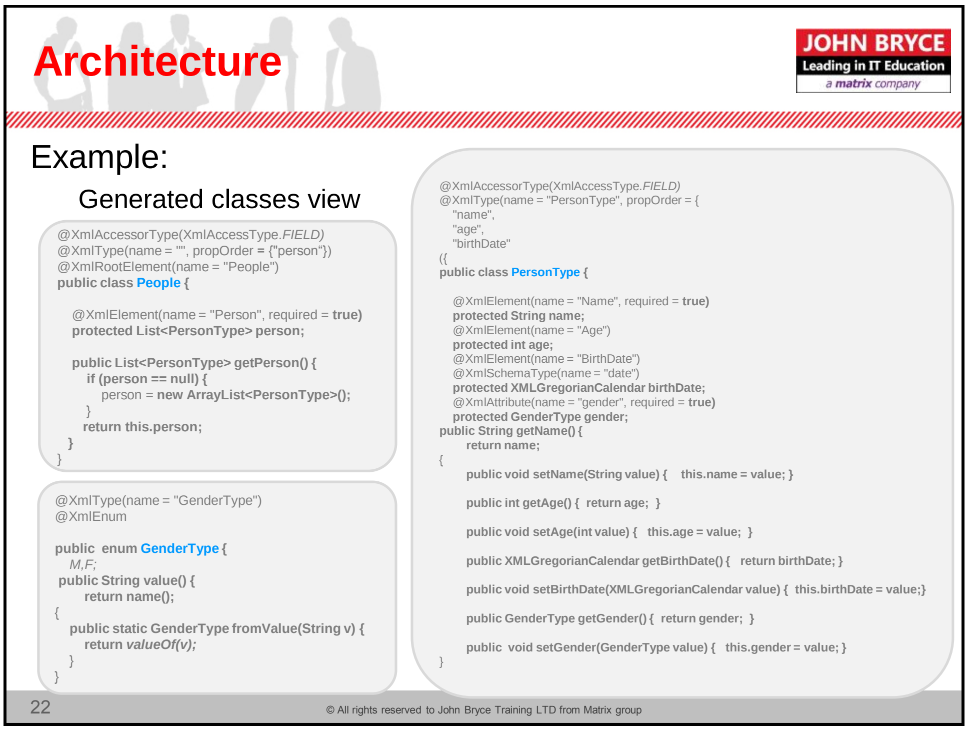
Marshalling

Export objects from memory into XML stream

Marshal operation takes:

- The root element of the content tree (objects)
- Output stream for writing XML

21



Architecture

Example:

Generated classes view

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {"person"})
@XmlRootElement(name = "People")
public class People {

    @XmlElement(name = "Person", required = true)
    protected List<PersonType> person;

    public List<PersonType> getPerson() {
        if (person == null) {
            person = new ArrayList<PersonType>();
        }
       return this.person;
   }
}
```

```
@XmlType(name = "GenderType")
@XmlEnum

public  enum GenderType {
    M,F;
 public String value() {
        return name();
{
    public static GenderType fromValue(String v) {
        return valueOf(v);
    }
}
```

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "PersonType", propOrder = {
    "name",
    "age",
    "birthDate"
({
public class PersonType {

    @XmlElement(name = "Name", required = true)
    protected String name;
    @XmlElement(name = "Age")
    protected int age;
    @XmlElement(name = "BirthDate")
    @XmlSchemaType(name = "date")
    protected XMLGregorianCalendar birthDate;
    @XmlAttribute(name = "gender", required = true)
    protected GenderType gender;
public String getName() {
        return name;
{
        public void setName(String value) {    this.name = value; }

        public int getAge() {  return age;  }

        public void setAge(int value) {   this.age = value;  }

        public XMLGregorianCalendar getBirthDate() {   return birthDate; }

        public void setBirthDate(XMLGregorianCalendar value) {  this.birthDate = value;}

        public GenderType getGender() {  return gender;  }

        public  void setGender(GenderType value) {   this.gender = value; }
}
```

22

Architecture

JOHN BRYCE Leading in IT Education a matrix company

Example:

Marshal and un-marshal with XML and JAXB

```
//un-marshal XML document
JAXBContext jc = JAXBContext.newInstance("core.people");
Unmarshaller unmarshaller = jc.createUnmarshaller();
People people=(People)unmarshaller.unmarshal(new File("c:/work/People.xml"));

//create new person data
PersonType person=new PersonType();
person.setName("Newbe");
person.setAge(1);
person.setGender(GenderType.M);
//add to people
List<PersonType> persons=people.getPerson();
persons.add(person);

//marshal back to XML document
Marshaller marshaller=jc.createMarshaller();

//optional - set marshaller validate structure
SchemaFactory sf = SchemaFactory.newInstance(
javax.xml.XMLConstants.W3C_XML_SCHEMA_NS_URI);
Schema schema = sf.newSchema(new File("http://...../PeopleSchema.xsd"));
marshaller.setSchema(schema);

marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
marshaller.marshal(people, new FileOutputStream("c:/work/PeopleUpdated.xml"));
```

Architecture

JOHN BRYCE Leading in IT Education a matrix company

XML vs. JSON

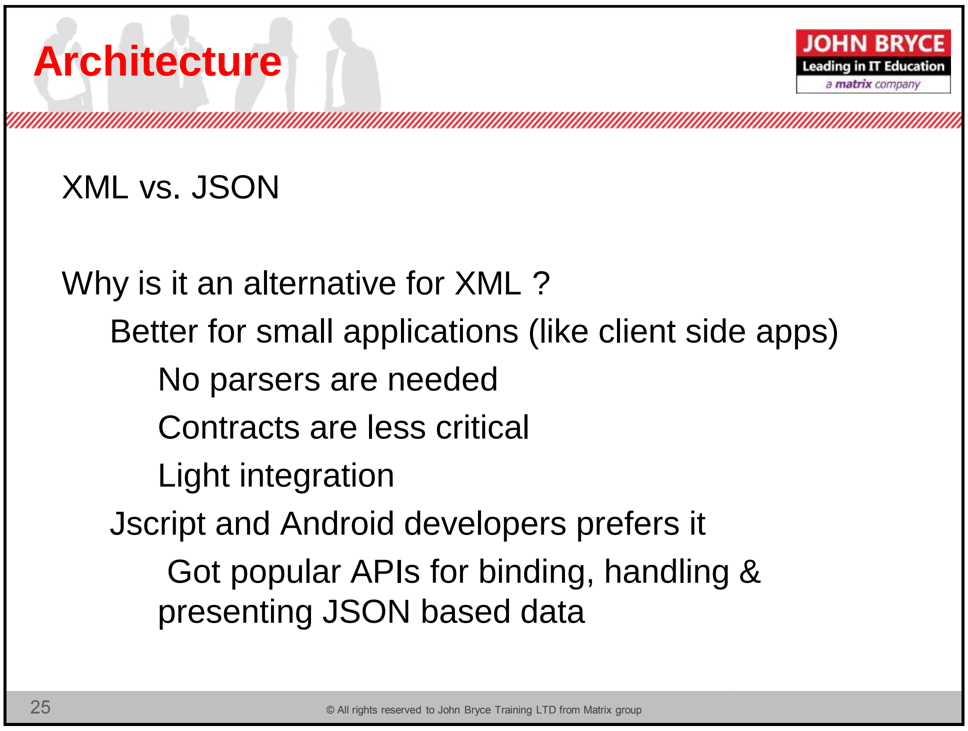
What is JSON ?

Java Script Object Notation

Also self-descriptive text based protocol

Used for marshalling and un-marshalling Jscript objects

```
{
 "people": [
          { "name": "David", "age": "20"},
          { "name": "Dana", "age": "25"},
          { "name": "Eve", age": "30"},
 ]
}
```

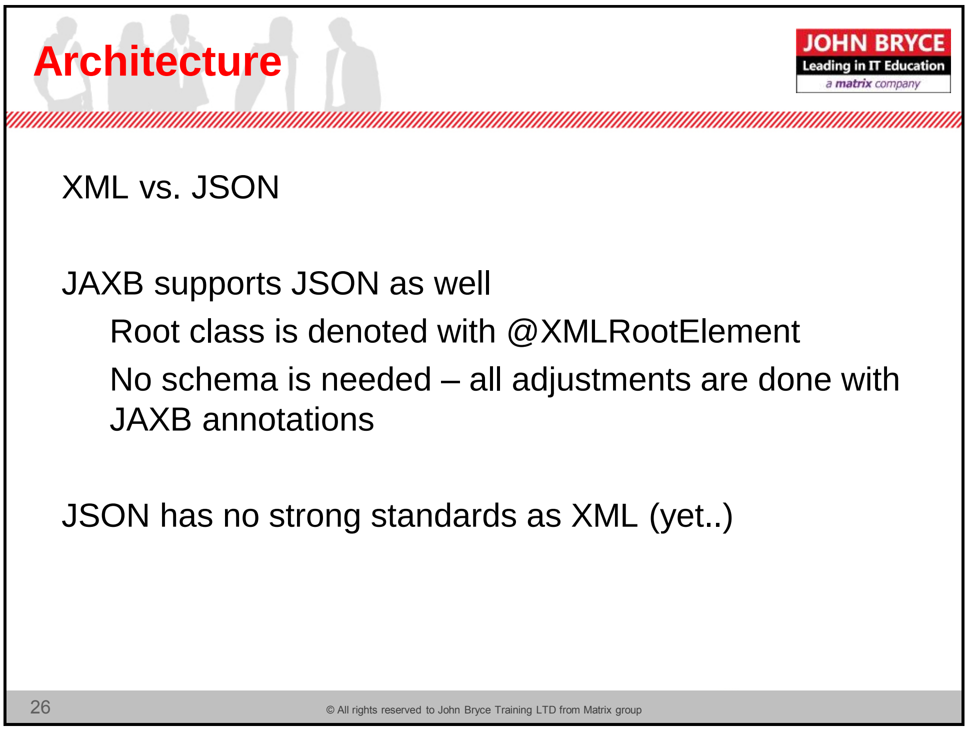


Architecture

XML vs. JSON

Why is it an alternative for XML ?
- Better for small applications (like client side apps)
  - No parsers are needed
  - Contracts are less critical
  - Light integration
- Jscript and Android developers prefers it
  - Got popular APIs for binding, handling & presenting JSON based data

25



Architecture

XML vs. JSON

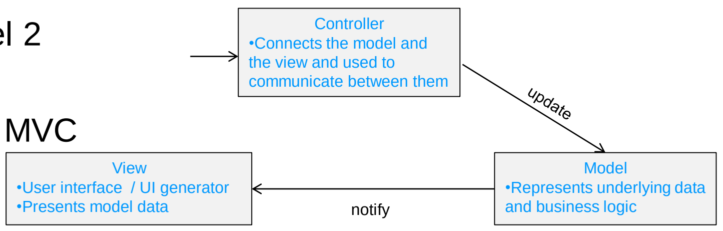
JAXB supports JSON as well
- Root class is denoted with @XMLRootElement
- No schema is needed – all adjustments are done with JAXB annotations

JSON has no strong standards as XML (yet..)

26

Architecture

MVC Model 2


Controller
•Connects the model and the view and used to communicate between them
update
View
•User interface / UI generator
•Presents model data
notify
Model
•Represents underlying data and business logic

Classic MVC

Designed to separate client flow and interaction from business model that serves the client request and from the final output.

Traditional MVC as used in web modules – MVC Model 2

Architecture

MVC Model 2

In JEE :

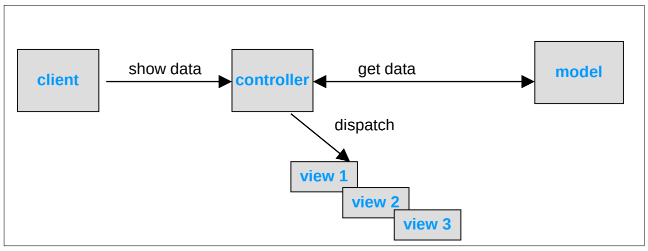
- Controller – Servlet (automated in JSF)
- View – JSP
- Model – EJB

J2EE Presentation Tier patterns

- **Service To Work –** MVC Model 2
- **Dispatcher View–** MVC Model 1

Architecture

Service To Work - MVC Model 2



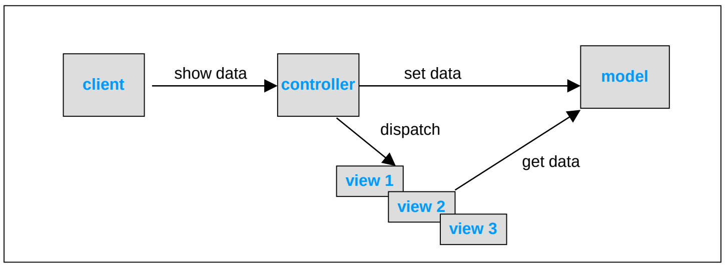
Better since:

- View and model communicated through value objects
- Trivial server side code is embedded in views

But, views are still a mix of server & client code...

Architecture

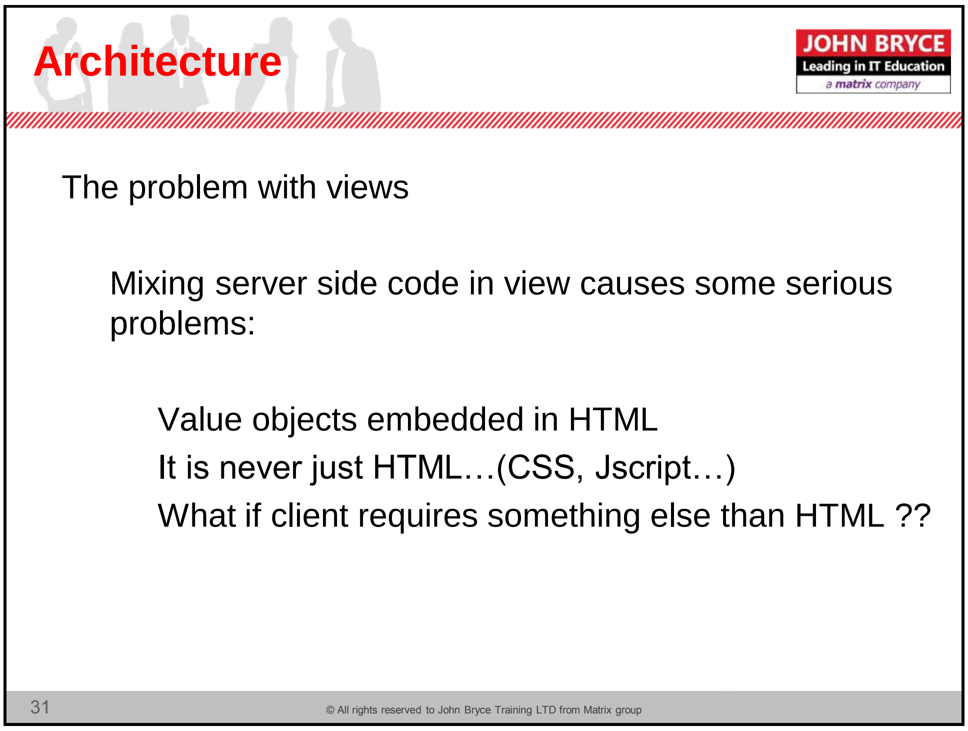
Dispatcher View model



Here, views are used also for ‘controlling’.

Tightly coupling between views and model

Can be considered for very simple modules



Architecture

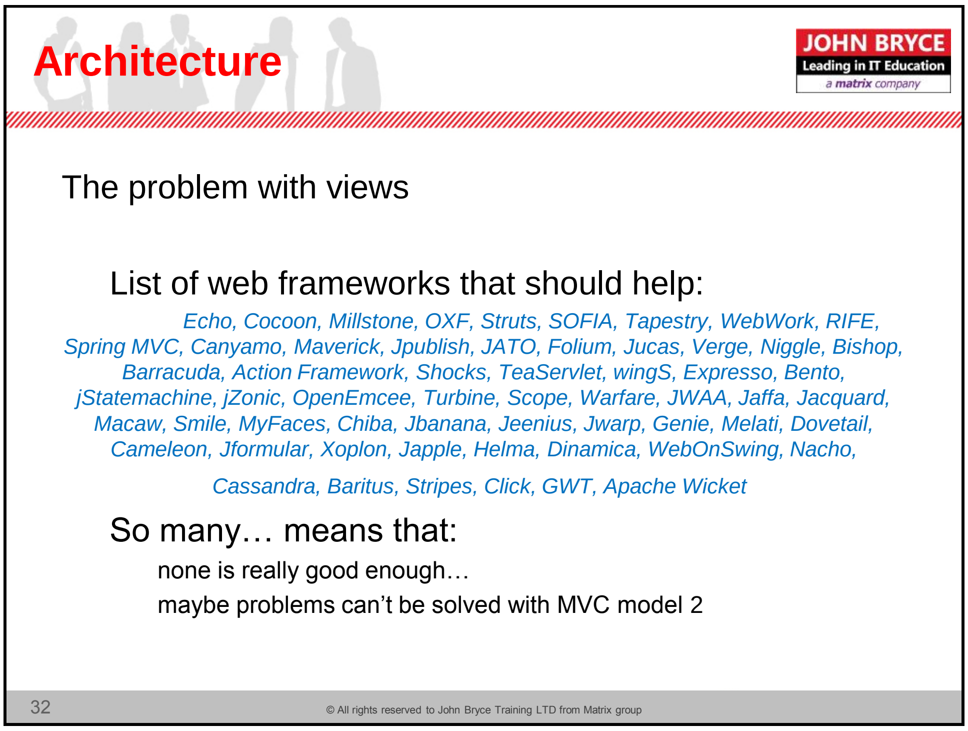
JOHN BRYCE
Leading in IT Education
a matrix company

The problem with views

Mixing server side code in view causes some serious problems:

- Value objects embedded in HTML
- It is never just HTML…(CSS, Jscript…)
- What if client requires something else than HTML ??

31



Architecture

JOHN BRYCE
Leading in IT Education
a matrix company

The problem with views

List of web frameworks that should help:

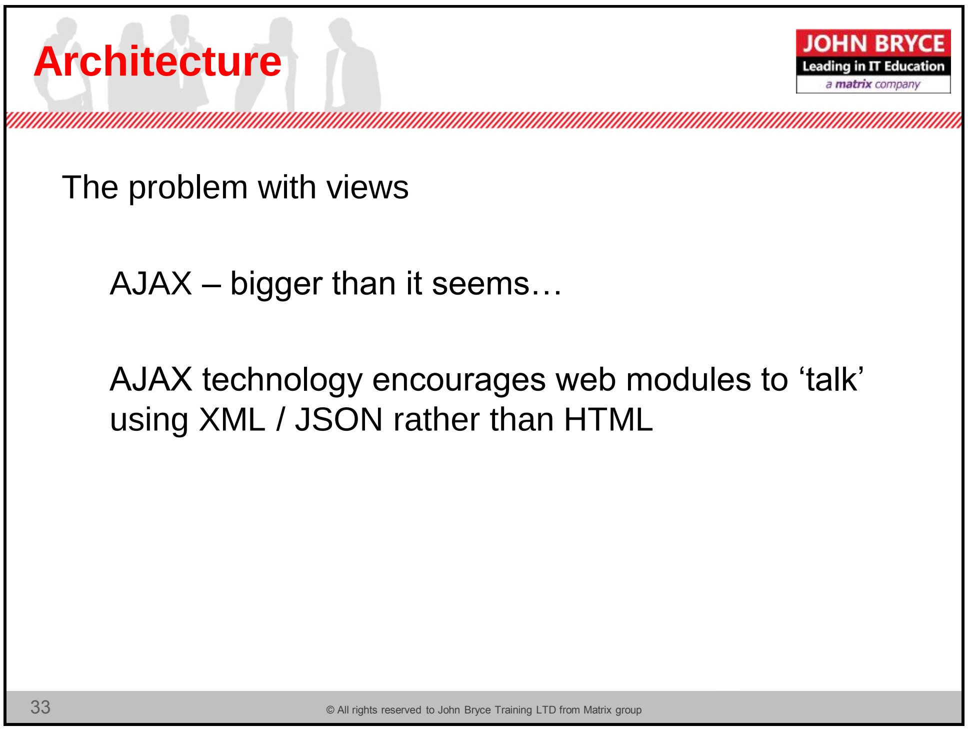
Echo, Cocoon, Millstone, OXF, Struts, SOFIA, Tapestry, WebWork, RIFE, Spring MVC, Canyamo, Maverick, Jpublish, JATO, Folium, Jucas, Verge, Niggle, Bishop, Barracuda, Action Framework, Shocks, TeaServlet, wingS, Expresso, Bento, jStatemachine, jZonic, OpenEmcee, Turbine, Scope, Warfare, JWAA, Jaffa, Jacquard, Macaw, Smile, MyFaces, Chiba, Jbanana, Jeenius, Jwarp, Genie, Melati, Dovetail, Cameleon, Jformular, Xoplon, Japple, Helma, Dinamica, WebOnSwing, Nacho,

Cassandra, Baritus, Stripes, Click, GWT, Apache Wicket

So many… means that:

- none is really good enough…
- maybe problems can't be solved with MVC model 2

32



Architecture

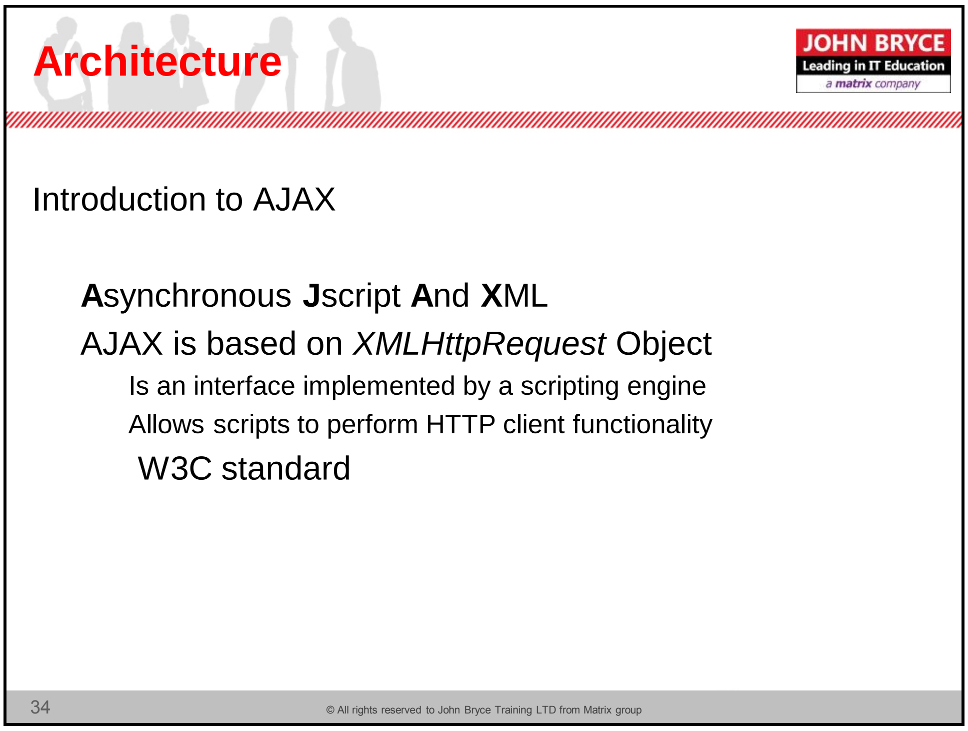
JOHN BRYCE
Leading in IT Education
a matrix company

The problem with views

AJAX – bigger than it seems…

AJAX technology encourages web modules to 'talk' using XML / JSON rather than HTML

33



Architecture

JOHN BRYCE
Leading in IT Education
a matrix company

Introduction to AJAX

Asynchronous **J**script **A**nd **X**ML

AJAX is based on *XMLHttpRequest* Object

Is an interface implemented by a scripting engine

Allows scripts to perform HTTP client functionality

W3C standard

34

Architecture

Introduction to AJAX

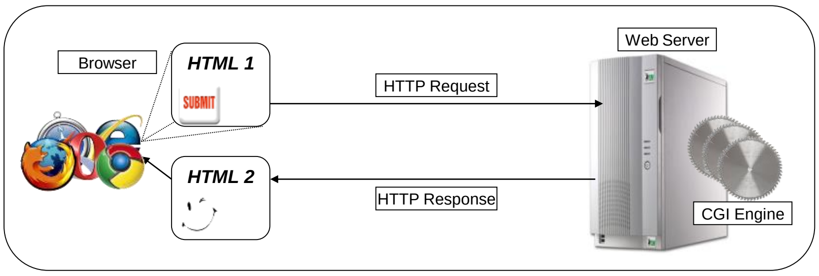
Classic way of interacting in web applications:

- Page by page
- Each page links or submits to another
- Static or dynamic content produced by the server
- Client side manipulation are done on downloaded data
- Most of client state is kept on server side

Architecture

Introduction to AJAX

View of classic architecture



Architecture

JOHN BRYCE Leading in IT Education a matrix company

Introduction to AJAX

AJAX way of interaction:

- Same page generates request(s) & processes responses
- Dynamic content handled also by the client
- Client downloads only the data he needs
- Client is notified asynchronously regarding data receiving

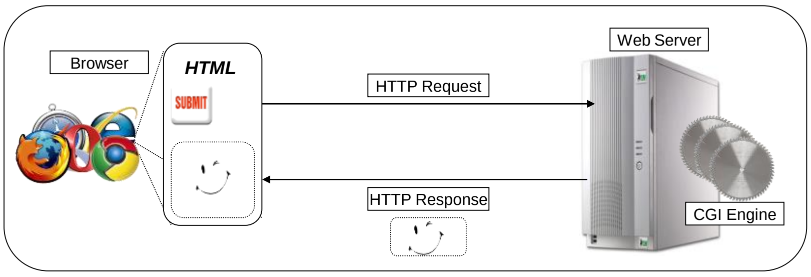
37 © All rights reserved to John Bryce Training LTD from Matrix group

Architecture

JOHN BRYCE Leading in IT Education a matrix company

Introduction to AJAX

View of AJAX architecture

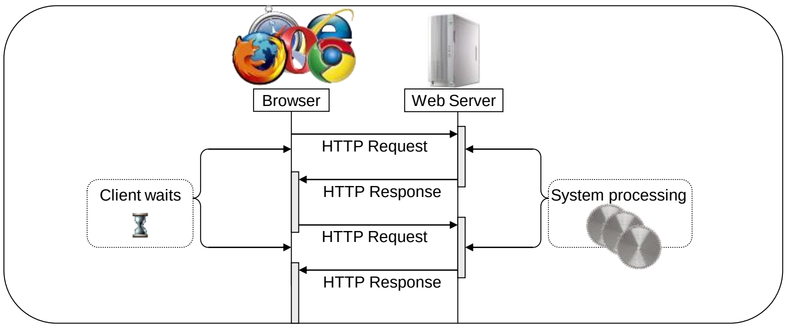


38 © All rights reserved to John Bryce Training LTD from Matrix group

Architecture

Introduction to AJAX

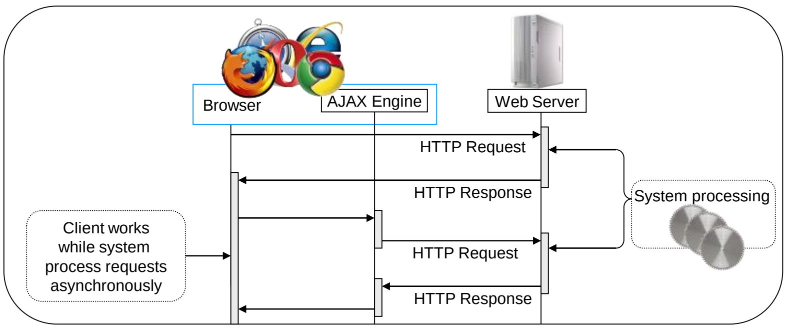
Classic interaction flow:



Architecture

Introduction to AJAX

AJAX interaction flow:



Architecture

JOHN BRYCE
Leading in IT Education
a matrix company

❑ Introduction to AJAX

url – the address of this ajax call. May target a Servlet or JSP. May send parameters just like any HTTP request

When a response is received stateCange() function will be asynchronously called

open() - Setting request data format:
- method (GET/POST)
- url
- asynchronous call – true enables it & is the default value

Send method takes a DOM Object. DOM object hosts XML documents and Fragments available via DOM API. Null value is also permitted, usually when string values are sent as a request header.

stateChange() method will be called asynchronously. Will be explained later.

```
function generateRequest()
{
    var url="http://localhost:8080/ajax";
    url+="?command=doIt";
    xmlhttp.onreadystatechange=stateChange;
    xmlhttp.open("GET",url,true);
    xmlhttp.send(null);
}

function stateChange()
{
    if (xmlhttp.readyState==4)
        // ...some code here...
    else
        alert("Problem retrieving XML data")
 }
…
```

Architecture

JOHN BRYCE
Leading in IT Education
a matrix company

Moving to single page applications

Using AJAX, web modules can focus on transferring data rather than view
- Client receives a single HTML loaded with Jscript functions & callbacks
- Jscript caller functions sends request data
- Jscript callback processes response and renders it to page

Finally !
- web modules input & output can be based on structured, self descriptive text formats
- Future non-HTML clients may use the same modules & data

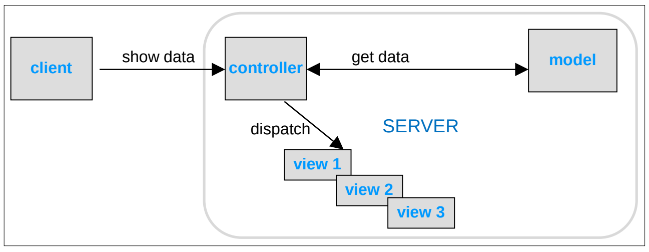
Architecture

JOHN BRYCE Leading in IT Education a matrix company

Moving to single page applications

MVC Model 2

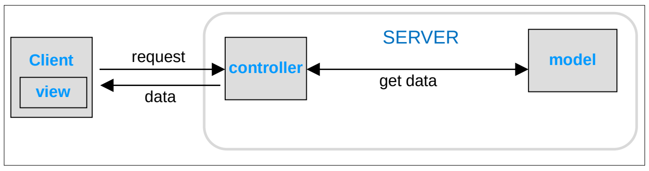
Views are generated on server side



The ‘new’ MVC

Views are handled by the client.

Communication between client and server is based on data



Architecture

JOHN BRYCE Leading in IT Education a matrix company

Future internet clients

Why is it so important to ‘talk’ via XML/JSON and not ‘draw’ HTMLs ?

Internet is much more than visiting web-sites…

Future client of the internet are not going to use keyboards and screens… HTML might be irrelevant

Architecture

JOHN BRYCE Leading in IT Education a matrix company

Future internet clients

- Phones & voice over IP networks
- Smart cards
- Chips
- Nanotechnology

Architecture

JOHN BRYCE Leading in IT Education a matrix company

Future internet clients

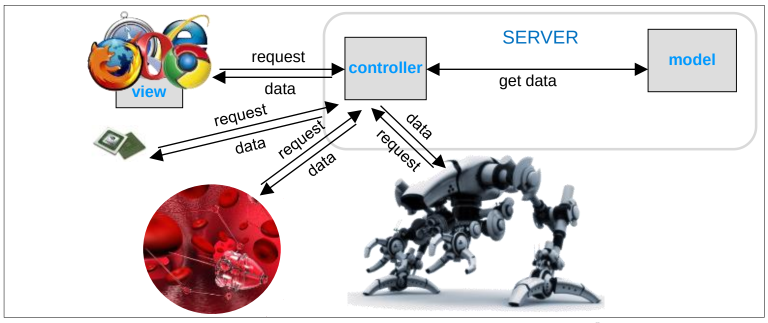
But the new ultimate client is US

- No hardware, no UI – just us
- Ability to share data directly from & to our brains

Architecture

Future internet clients

The 'new' MVC



RPC

Architecture & terms
RPC in Java & JEE
RPC Framework requirements
XML for RPC – Web services

RPC

Architecture & Terms

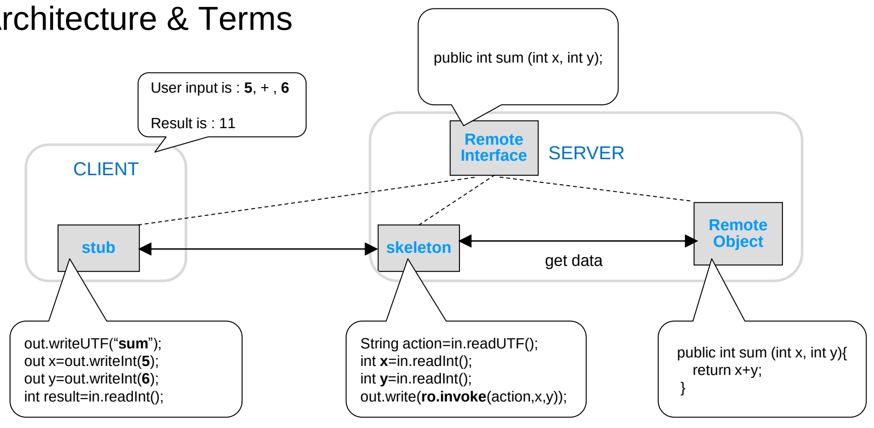
Remote Procedure Call

- Client invokes method on a Remote Object over a network
- Client obeys a contract which is the Remote Interface
- Remote object is a resource
- Remote method is a service

- In order to communicate both client & server uses sockets
  - Socket communication is determined according to the remote interface
  - Stub - Client side socket
  - Skeleton – server side socket that is used as a proxy to the remote object

RPC

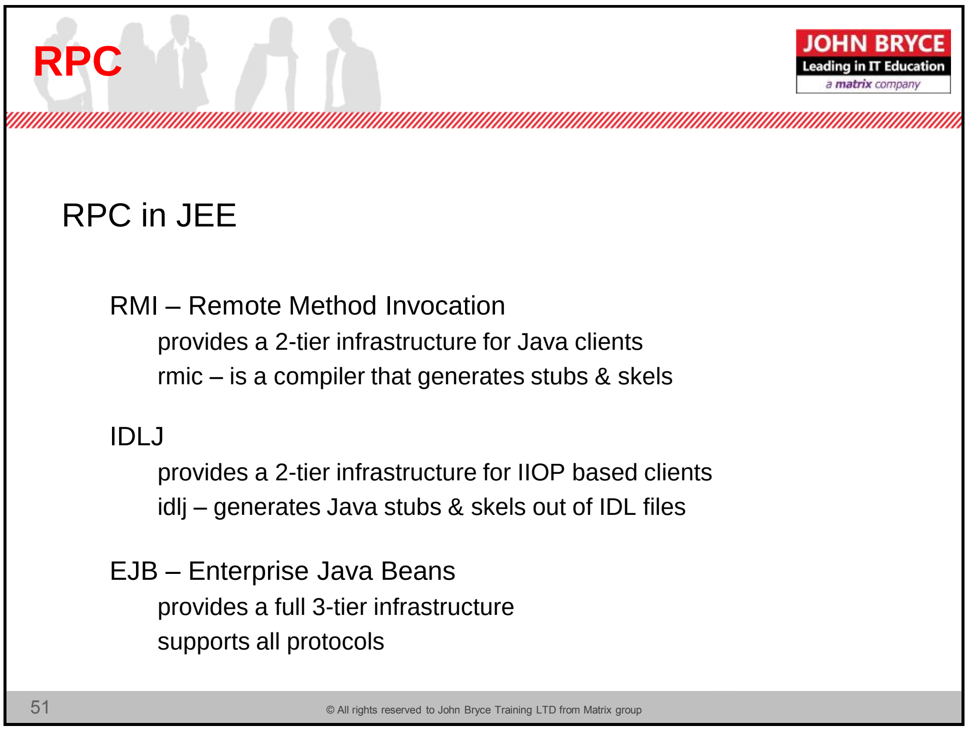
Architecture & Terms



```
out.writeUTF("sum");
out x=out.writeInt(5);
out y=out.writeInt(6);
int result=in.readInt();
```

```
String action=in.readUTF();
int x=in.readInt();
int y=in.readInt();
out.write(ro.invoke(action,x,y));
```

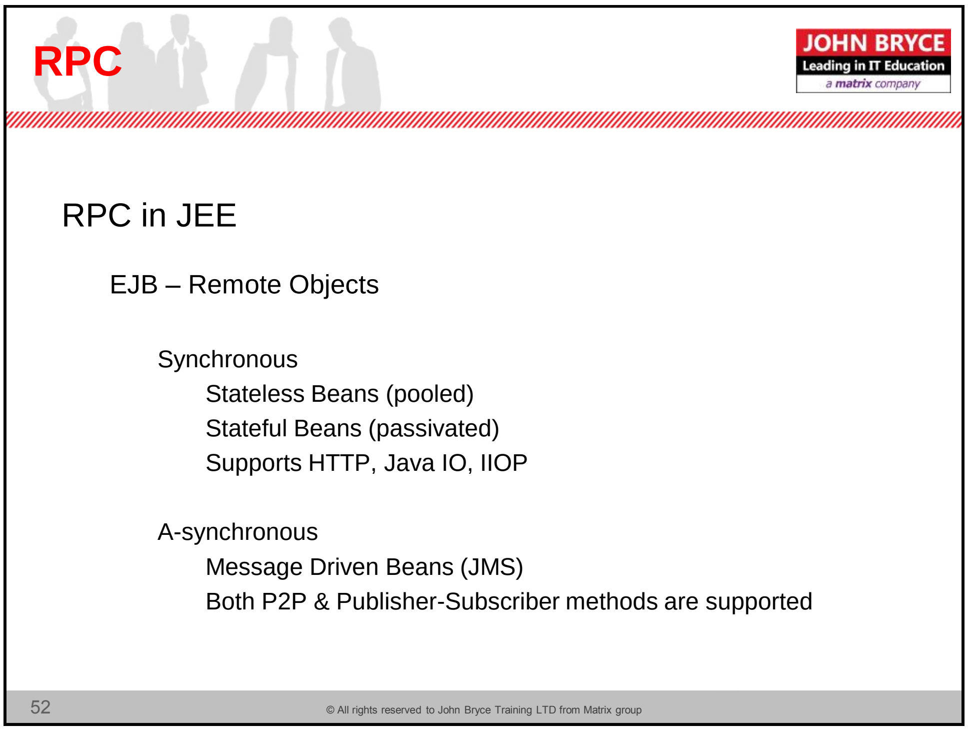
```
public int sum (int x, int y){
    return x+y;
}
```



RPC

RPC in JEE

- RMI – Remote Method Invocation
  - provides a 2-tier infrastructure for Java clients
  - rmic – is a compiler that generates stubs & skels
- IDLJ
  - provides a 2-tier infrastructure for IIOP based clients
  - idlj – generates Java stubs & skels out of IDL files
- EJB – Enterprise Java Beans
  - provides a full 3-tier infrastructure
  - supports all protocols



RPC

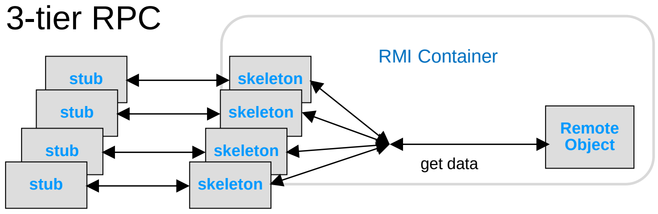
RPC in JEE

- EJB – Remote Objects
  - Synchronous
    - Stateless Beans (pooled)
    - Stateful Beans (passivated)
    - Supports HTTP, Java IO, IIOP
  - A-synchronous
    - Message Driven Beans (JMS)
    - Both P2P & Publisher-Subscriber methods are supported

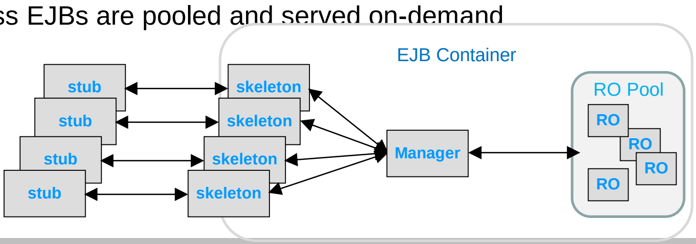
RPC

RPC in JEE

2-tier vs. 3-tier RPC



Stateless EJBs are pooled and served on-demand



RPC

RPC Framework Requirements

For server development

- Map services to generate a contract
- Expose the contract
- Instantiate & publish the resource (or resource pool)
- Create skeleton when requested

For client development

- Generate stub according to a given contract

XML for RPC – Web Services

Main goal of XML is for application integration

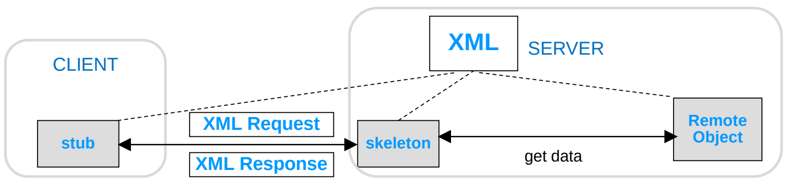
If the contract is in XML format it can be:

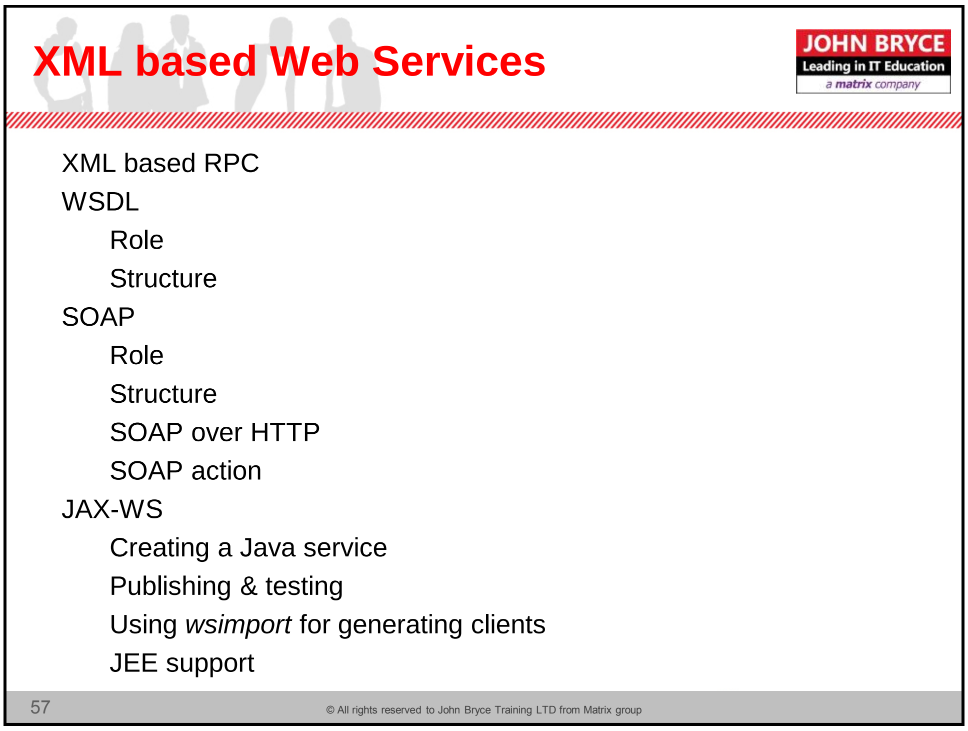
- describing services written in any language
- used by any client

If the stubs & skels will ‘talk’ via XML:

- each may be written in a different language
- xsd types can be used to describe primitives & objects

XML for RPC – Web Services



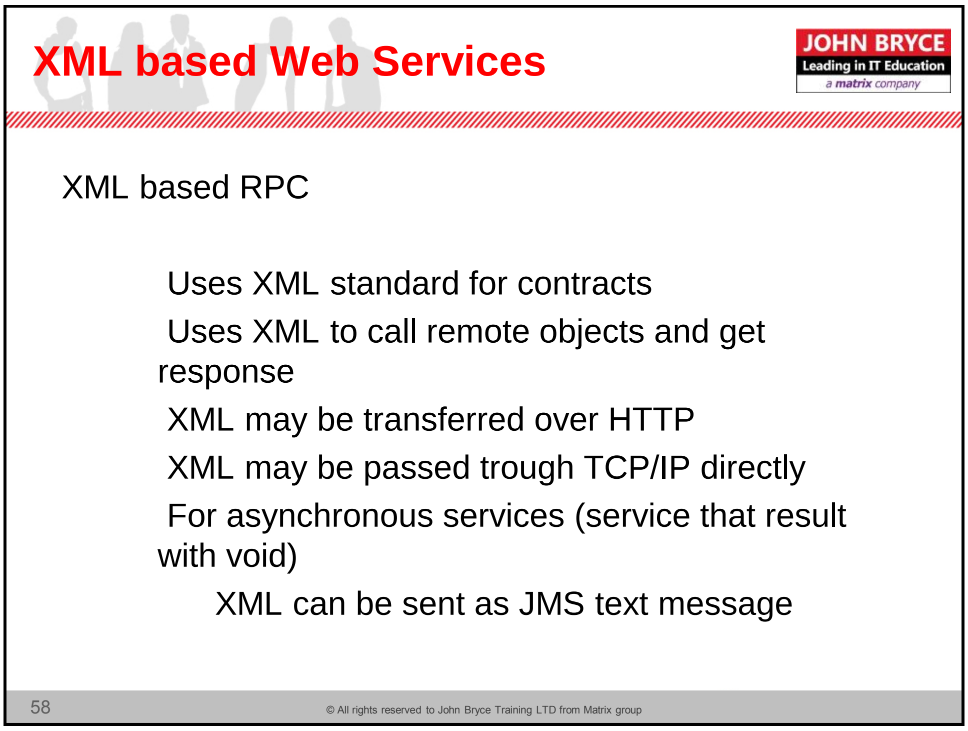


XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

- XML based RPC
- WSDL
  - Role
  - Structure
- SOAP
  - Role
  - Structure
  - SOAP over HTTP
  - SOAP action
- JAX-WS
  - Creating a Java service
  - Publishing & testing
  - Using *wsimport* for generating clients
  - JEE support

57



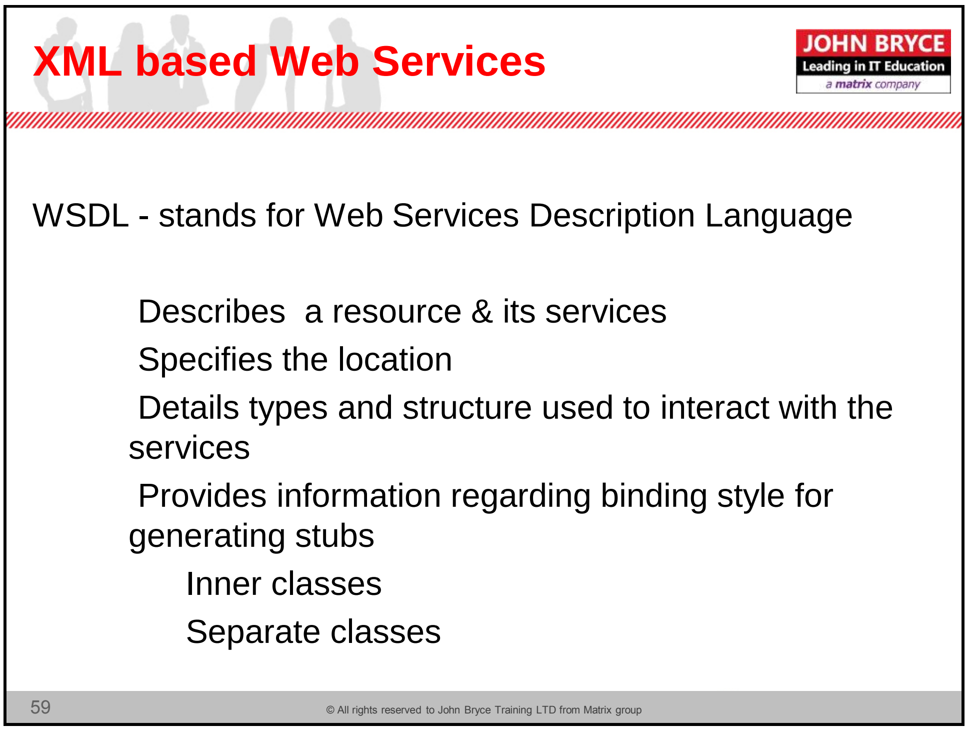
XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

XML based RPC

- Uses XML standard for contracts
- Uses XML to call remote objects and get response
- XML may be transferred over HTTP
- XML may be passed trough TCP/IP directly
- For asynchronous services (service that result with void)
  - XML can be sent as JMS text message

58



XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

WSDL - stands for Web Services Description Language

Describes a resource & its services

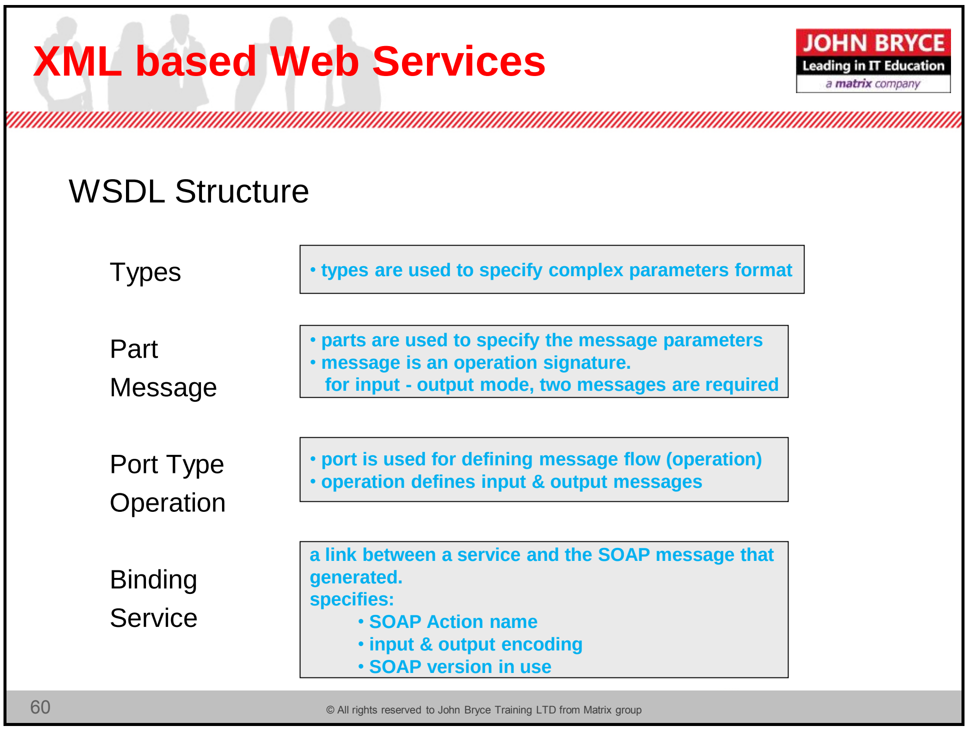
Specifies the location

Details types and structure used to interact with the services

Provides information regarding binding style for generating stubs

- Inner classes
- Separate classes

59



XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

WSDL Structure

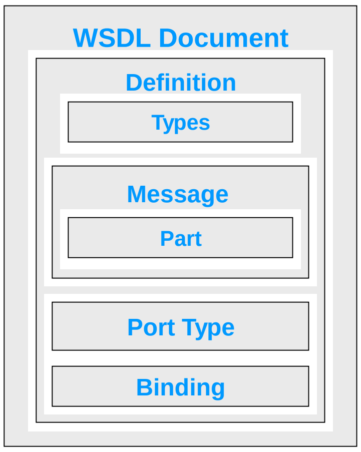
Types	• types are used to specify complex parameters format
Part Message	• parts are used to specify the message parameters • message is an operation signature. for input - output mode, two messages are required
Port Type Operation	• port is used for defining message flow (operation) • operation defines input & output messages
Binding Service	a link between a service and the SOAP message that generated. specifies: • SOAP Action name • input & output encoding • SOAP version in use

60

XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

WSDL Structure



XML based Web Services

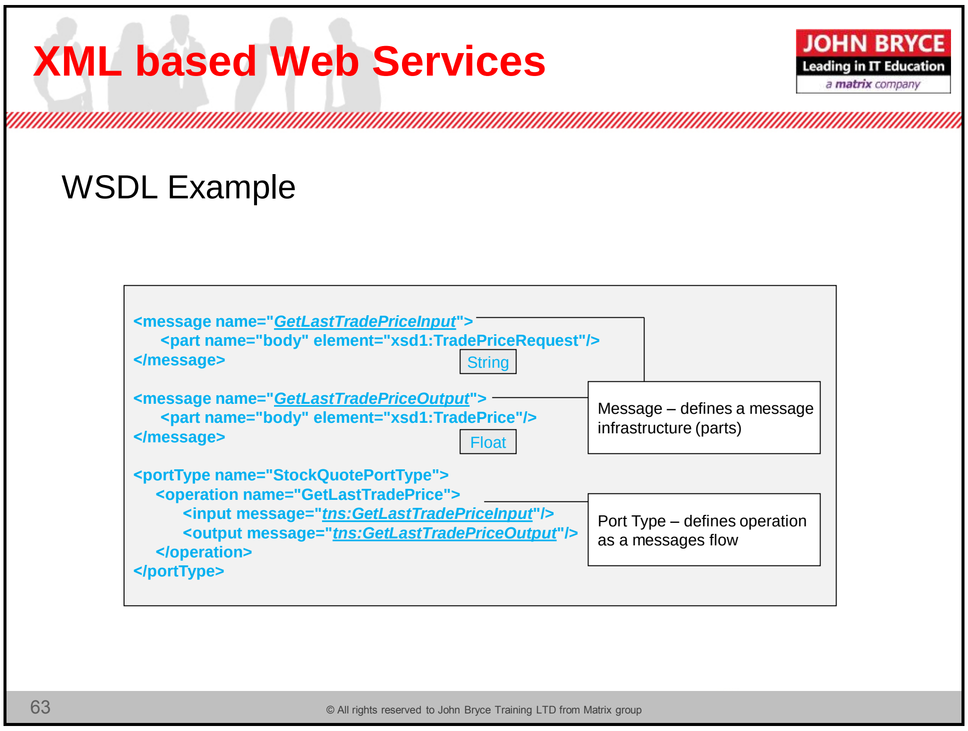
JOHN BRYCE
Leading in IT Education
a matrix company

WSDL Example

This WSDL defines the following service:
• client sends a stoke symbol in string format
• client gets the stoke value in float format

```
<?xml version="1.0"?>
<definitions name="StockQuote"
          targetNamespace="http://example.com/stockquote.wsdl"
          xmlns:tns="http://example.com/stockquote.wsdl"
          xmlns:xsd1="http://example.com/stockquote.xsd"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns="http://schemas.xmlsoap.org/wsdl/">

< types>
   <schema targetNamespace="http://example.com/stockquote.xsd"
         xmlns="http://www.w3.org/2000/10/XMLSchema">
       <element name="TradePriceRequest">  String
           <complexType>
             <all> <element name="tickerSymbol" type="string"/> </all>
           </complexType>
       </element>
       <element name="TradePrice">  Float
           <complexType>
              <all><element name="price" type="float"/></all>
           </complexType>
        </element>
    </schema>
</types>…
```

Types – defines the request & response parameters format



XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

WSDL Example

```
<message name="GetLastTradePriceInput">
     <part name="body" element="xsd1:TradePriceRequest"/>
</message>
```

String

```
<message name="GetLastTradePriceOutput">
     <part name="body" element="xsd1:TradePrice"/>
</message>
```

Float

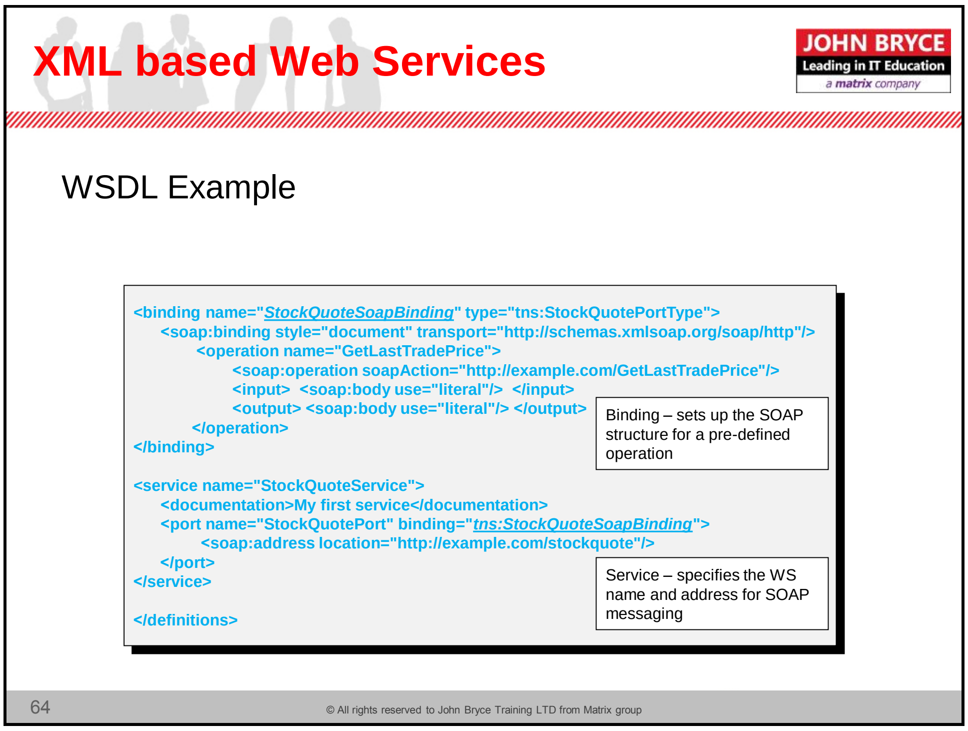
```
<portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
         <input message="tns:GetLastTradePriceInput"/>
         <output message="tns:GetLastTradePriceOutput"/>
    </operation>
</portType>
```

Message – defines a message infrastructure (parts)

Port Type – defines operation as a messages flow

63

© All rights reserved to John Bryce Training LTD from Matrix group



XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

WSDL Example

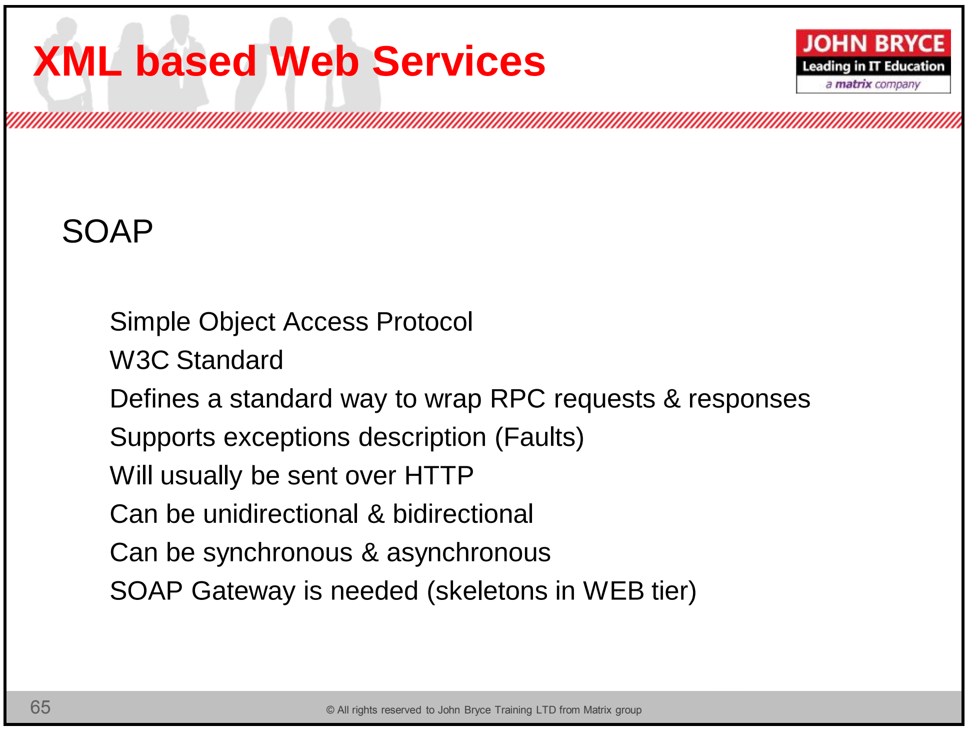
```
<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
     <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
           <operation name="GetLastTradePrice">
                 <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
                 <input>  <soap:body use="literal"/>  </input>
                 <output> <soap:body use="literal"/> </output>
           </operation>
</binding>

<service name="StockQuoteService">
     <documentation>My first service</documentation>
     <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
            <soap:address location="http://example.com/stockquote"/>
     </port>
</service>

</definitions>
```

Binding – sets up the SOAP structure for a pre-defined operation

Service – specifies the WS name and address for SOAP messaging

64

© All rights reserved to John Bryce Training LTD from Matrix group

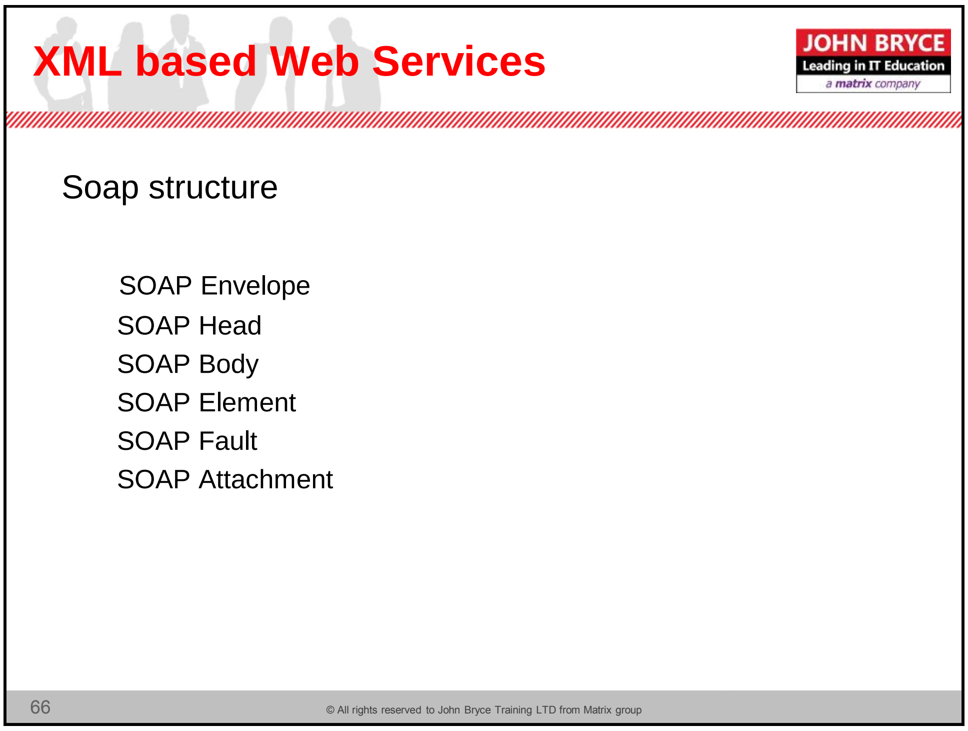


XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

SOAP

- Simple Object Access Protocol
- W3C Standard
- Defines a standard way to wrap RPC requests & responses
- Supports exceptions description (Faults)
- Will usually be sent over HTTP
- Can be unidirectional & bidirectional
- Can be synchronous & asynchronous
- SOAP Gateway is needed (skeletons in WEB tier)



XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

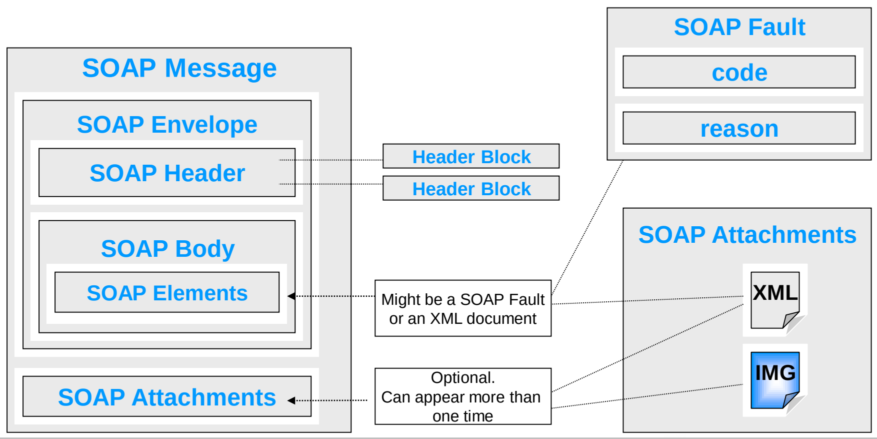
Soap structure

- SOAP Envelope
- SOAP Head
- SOAP Body
- SOAP Element
- SOAP Fault
- SOAP Attachment

XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

Soap structure schema



XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

- SOAP over HTTP Request example

```
HTTP 1.0 POST
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI“
```

POST method - allows great amount of data upload

SOAP Header that provides another checking mechanism

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
               SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <SOAP-ENV:Body>
           <m:GetLastTradePrice xmlns:m="Some-URI">
                 <symbol>DIS</symbol>
           </m:GetLastTradePrice>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

SOAP & WSDL binding styles

Generating stubs to use SOAP is done in 2 different styles

- SOAP-RPC
- SOAP-DOCUMENTED

Style is specified in WSDL

XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

SOAP- RPC Style

WSDL sets all complex types internally

```
<types>
 <schema targetNamespace="http://example.com/stockquote.xsd"
         xmlns="http://www.w3.org/2000/10/XMLSchema">
        <element name="TradePriceRequest">
            <complexType>
               <all> <element name="tickerSymbol" type="string"/> </all>
            </complexType>
        </element>
        <element name="TradePrice">
            <complexType>
               <all><element name="price" type="float"/></all>
            </complexType>
         </element>
     </schema>
</types>
```

```
<!-- no complex types are used, only
     simple types exists in XSD, like :
     xsd:int, xsd:string, xsd:date, etc.
-->

<types/>
```

In Java – when generating RPC based stub, all types will be generated as stubs inner classes

XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

SOAP- RPC Style

When no complex types exists prefer RPC:

Relevant when we use types that can be mapped to schema types directly

Like: xsd:int, xsd:string, xsd:date, etc.

```
<types/>
```

XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

SOAP- DOCUMENTED Style

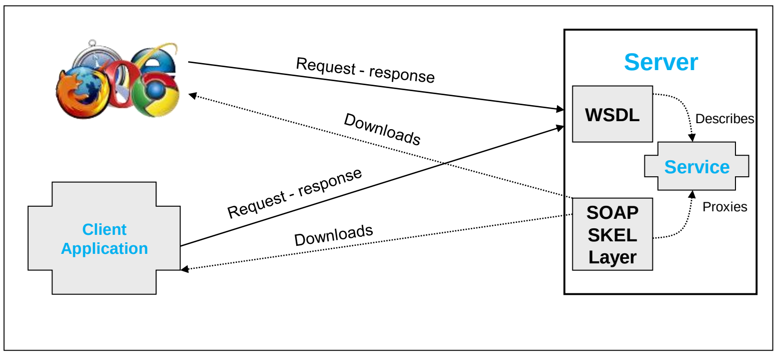
WSDL holds references to external XSD schemas that defines all complex types

```
<types>
   <xsd:schema>
            <xsd:import schemaLocation="http://localhost:8080/calc?xsd=1" namespace="http://core/"/>
   </xsd:schema>
</types>
```

In Java – when generating DUCUMENTED based stub, all types will be generated as stand alone classes

XML based Web Services

- XML based RPC



XML based Web Services

JAX – WS

Java API for XML – Web Services
Provides

Annotations for WS mapping
Tool for client code generation
Embedded HTTP server

Light-weight
Accessible via java object
Can be started programmatically
Endpoint.publish("……..")
Listens to port 8080 by default

XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

Who comes first ? Java or WSDL ?

Java first
- First we code our Java model, than we generate WSDL to describe our services
- Usually, we expose existing business logic
- Usually default mappings to WSDL works fine

Contract first
- First we generate a WSDL, than we create a Java based interface & implement it
- Relevant when:
  - We are forced to work according to WSDL
  - Java auto generated WSDL is too complex or not optimized

Both methods are supported in JAX-WS

XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

Mapping class methods via JAX-WS annotations

```
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;

@WebService
@SOAPBinding(style=SOAPBinding.Style.RPC)
public class Business {
        @WebMethod
        public int sum(int x,int y){
                return x+y;
        }
}
```

XML based Web Services

- *@WebService* annotation
  - Used for classes
  - Important attributes:
    - *name* – specifies the name of the web-service when pointing to the WSDL
    - default value – the name of the mapped class of interface
    - *wsdlLocation*– the location of the WSDL - when not automatically generated
    - default value – *http://<host>:<port>/<app dir>/<service name>?wsdl*
    - example: *http://localhost:8080/brokerApp/stockWS?wsdl*
    - *endpointInterface*– specifies the name of business interface
    - default – the class itself if POJO or remote interface for Remote Objects

XML based Web Services

- *@WebMethod* annotation
  - Is used at the method level
    - denotes method to be included in the WSDL as an operation
  - Attributes:
    - *operationName*– specifies the name of the web-service operation
      default value – the actual name of the mapped method
    - *exclude*– a flag indicates if the method will be documented in WSDL or not
      default value - true
    - *action*– the SOAP action header that is bounded to this operation
      default – none

SOAP Action – A HTTP header that allows the system to map a WS client call to the desired destination efficiently. Without SOAP Action the SOAP message has to be parsed & examined in order to be correctly delegated to destination.

XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

Main JAX-WS Annotations

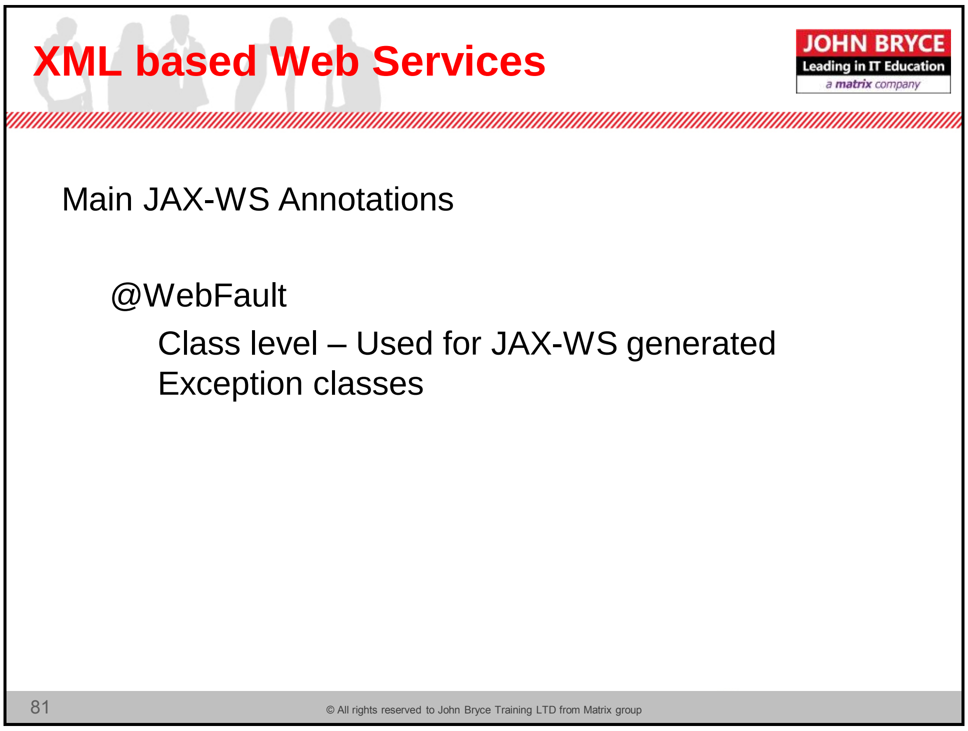
- @WebParam
  - Method param level – sets a parameter of an operation
  - Defines – name, namespace (for DOCUMENT style) and mode (IN / OUT / INOUT)
- @WebResult
  - Method param level – sets a result of an operation
  - Defines – name, namespace (for DOCUMENT style)

XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

Main JAX-WS Annotations

- @SOAPBinding
  - Class level – Sets SOAP-WSDL binding
  - Defines – DOCUMENT(default) / RPC SOAP style and encoding to use:
    - LITERAL (default) – the XML in the body is taken as is
    - ENCODED – currently not supported
- @RequestWrapper
- @ResponeWrapper
  - Sets a Java class to listen to ingoing and outgoing calls



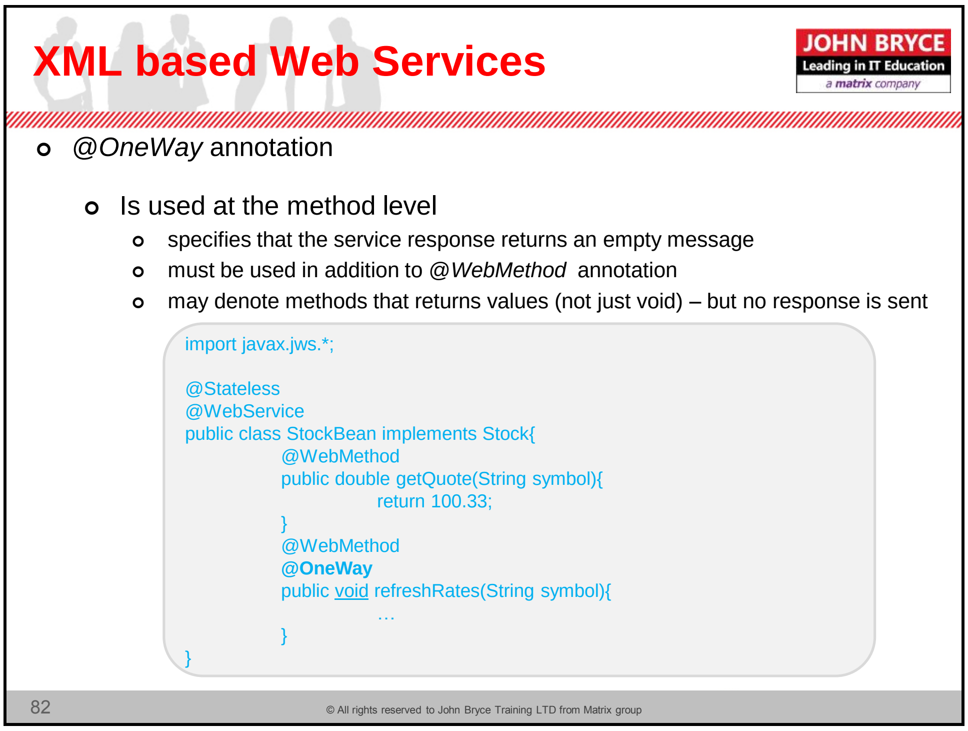
XML based Web Services

JOHN BRYCE Leading in IT Education *a matrix company*

Main JAX-WS Annotations

@WebFault

Class level – Used for JAX-WS generated Exception classes



XML based Web Services

JOHN BRYCE Leading in IT Education *a matrix company*

- *@OneWay* annotation
  - Is used at the method level
    - specifies that the service response returns an empty message
    - must be used in addition to *@WebMethod* annotation
    - may denote methods that returns values (not just void) – but no response is sent

```
import javax.jws.*;

@Stateless
@WebService
public class StockBean implements Stock{
            @WebMethod
            public double getQuote(String symbol){
                        return 100.33;
            }
            @WebMethod
            @OneWay
            public void refreshRates(String symbol){
                        …
            }
}
```

XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

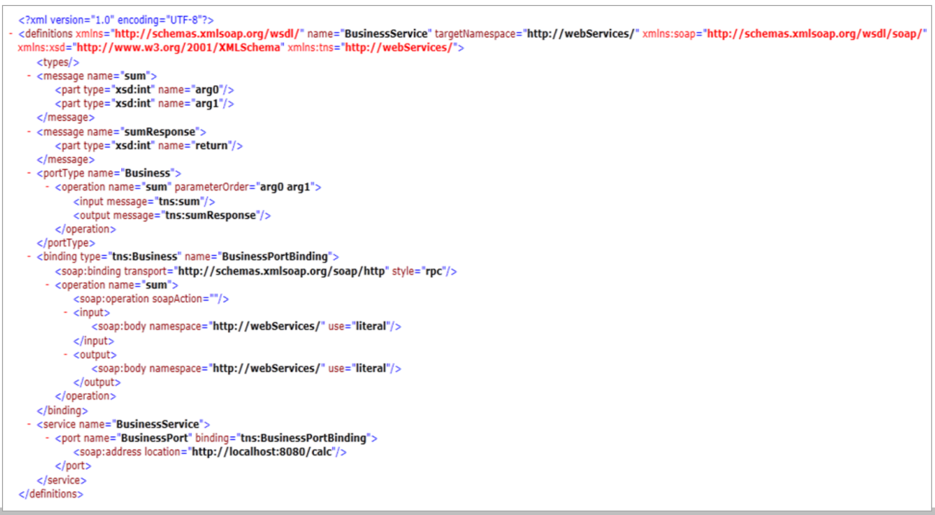
Publishing as WS with a given endpoint

```
import javax.xml.ws.Endpoint;

public class Publisher {

                public static void main(String[] args) {
                                Endpoint.publish("http://localhost:8080/calc",new Business());

                }

}
```

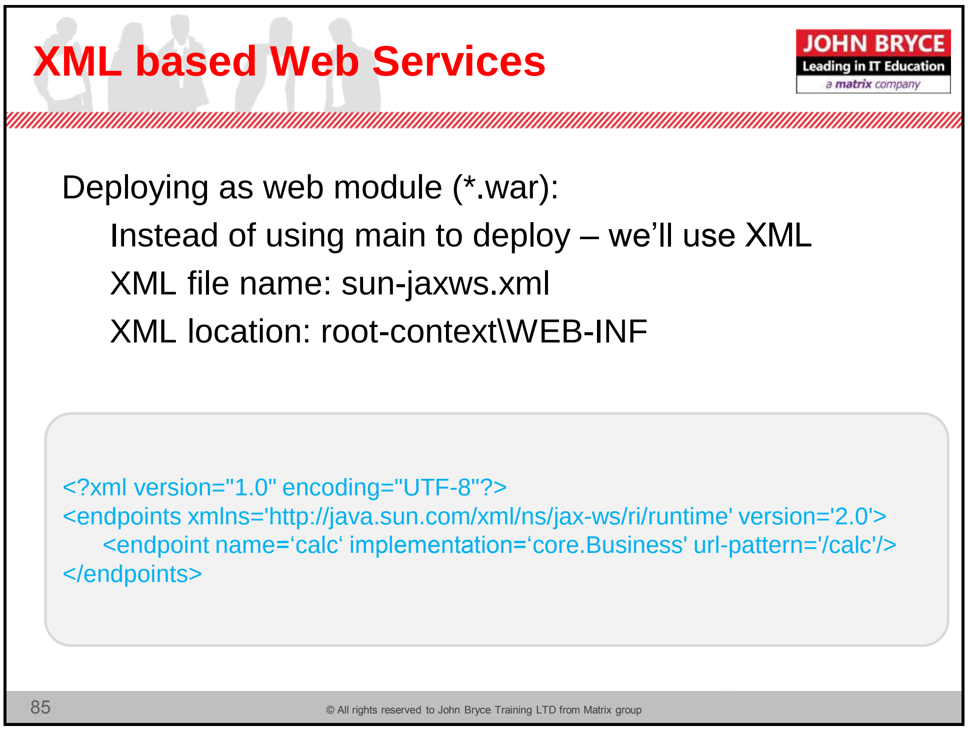
XML based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

WSDL View



```
  <?xml version="1.0" encoding="UTF-8"?>
- <definitions xmlns="http://schemas.xmlsoap.org/wsdl/" name="BusinessService" targetNamespace="http://webServices/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://webServices/">
    <types/>
  - <message name="sum">
      <part type="xsd:int" name="arg0"/>
      <part type="xsd:int" name="arg1"/>
    </message>
  - <message name="sumResponse">
      <part type="xsd:int" name="return"/>
    </message>
  - <portType name="Business">
    - <operation name="sum" parameterOrder="arg0 arg1">
        <input message="tns:sum"/>
        <output message="tns:sumResponse"/>
      </operation>
    </portType>
  - <binding type="tns:Business" name="BusinessPortBinding">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    - <operation name="sum">
        <soap:operation soapAction=""/>
      - <input>
          <soap:body namespace="http://webServices/" use="literal"/>
        </input>
      - <output>
          <soap:body namespace="http://webServices/" use="literal"/>
        </output>
      </operation>
    </binding>
  - <service name="BusinessService">
    - <port name="BusinessPort" binding="tns:BusinessPortBinding">
        <soap:address location="http://localhost:8080/calc"/>
      </port>
    </service>
  </definitions>
```



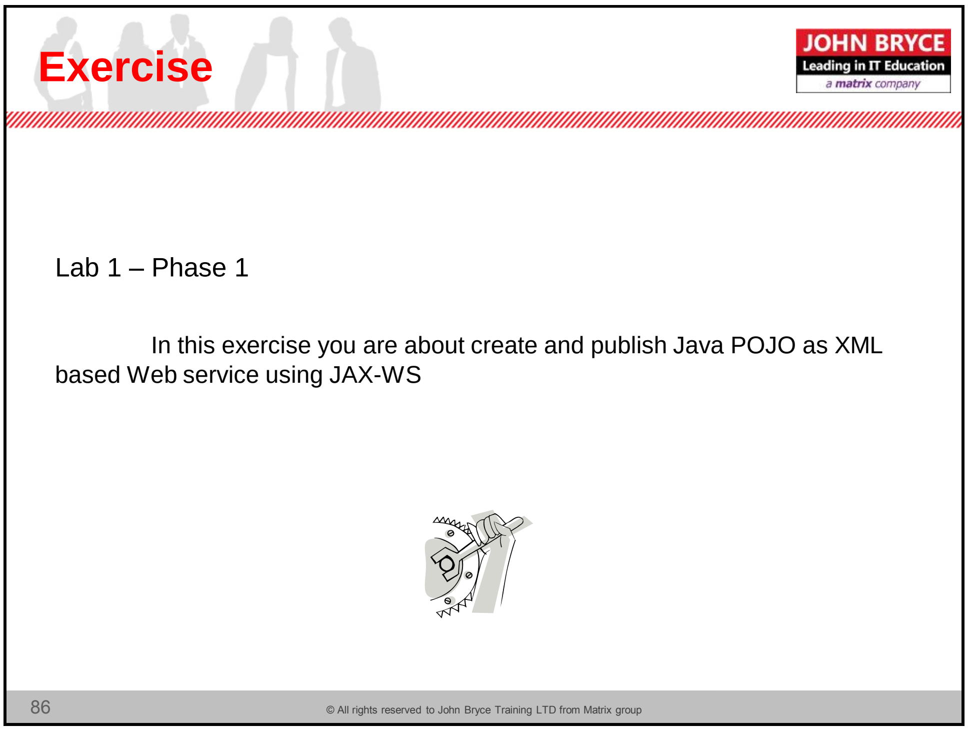
XML based Web Services

Deploying as web module (*.war):

- Instead of using main to deploy – we'll use XML
- XML file name: sun-jaxws.xml
- XML location: root-context\WEB-INF

```
<?xml version="1.0" encoding="UTF-8"?>
<endpoints xmlns='http://java.sun.com/xml/ns/jax-ws/ri/runtime' version='2.0'>
    <endpoint name=‘calc‘ implementation=‘core.Business' url-pattern='/calc'/>
</endpoints>
```

85

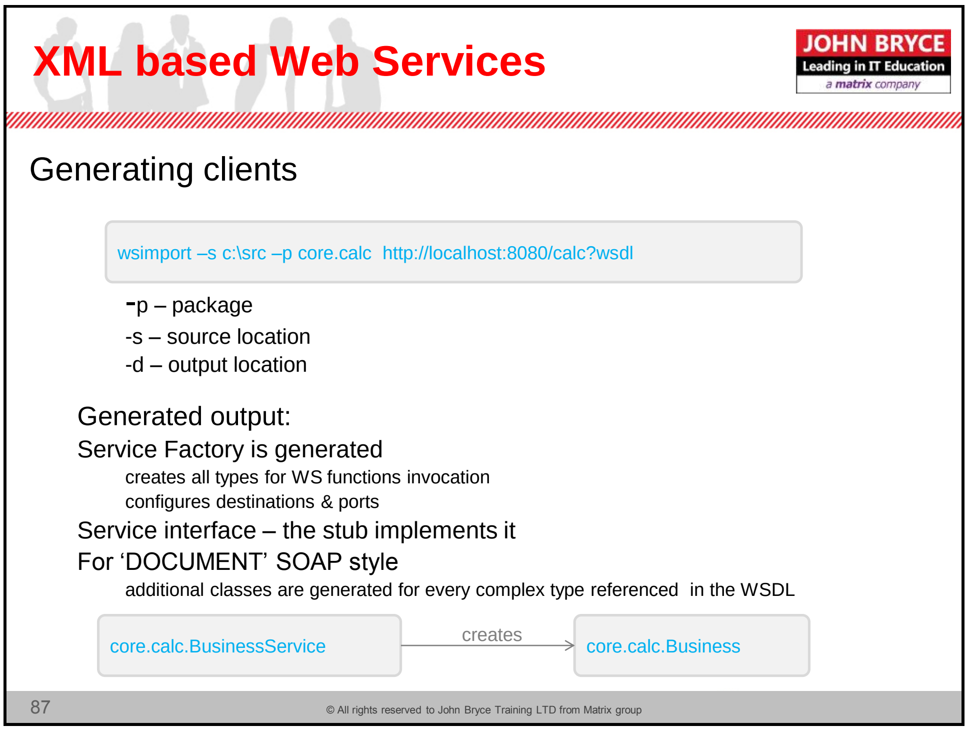


Exercise

Lab 1 – Phase 1

In this exercise you are about create and publish Java POJO as XML based Web service using JAX-WS

86



XML based Web Services

JOHN BRYCE Leading in IT Education a matrix company

Generating clients

```
wsimport –s c:\src –p core.calc  http://localhost:8080/calc?wsdl
```

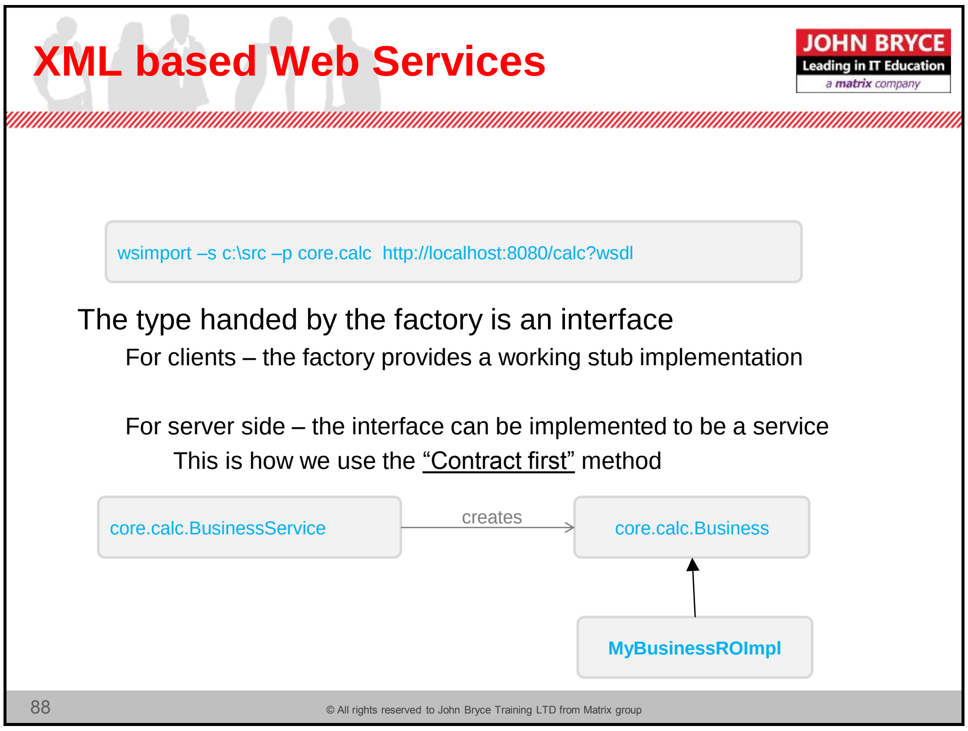
-p – package
-s – source location
-d – output location

Generated output:
Service Factory is generated
- creates all types for WS functions invocation
- configures destinations & ports

Service interface – the stub implements it
For ‘DOCUMENT’ SOAP style
- additional classes are generated for every complex type referenced in the WSDL

$$\begin{tikzcd}\text{core.calc.BusinessService} \arrow[r, "\text{creates}"] & \text{core.calc.Business}\end{tikzcd}$$

87



XML based Web Services

JOHN BRYCE Leading in IT Education a matrix company

```
wsimport –s c:\src –p core.calc  http://localhost:8080/calc?wsdl
```

The type handed by the factory is an interface
- For clients – the factory provides a working stub implementation

- For server side – the interface can be implemented to be a service
  - This is how we use the <u>“Contract first”</u> method

$$\begin{tikzcd}\text{core.calc.BusinessService} \arrow[r, "\text{creates}"] & \text{core.calc.Business} \\ & \textbf{MyBusinessROImpl} \arrow[u]\end{tikzcd}$$

88

XML based Web Services

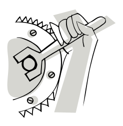
Client code example:

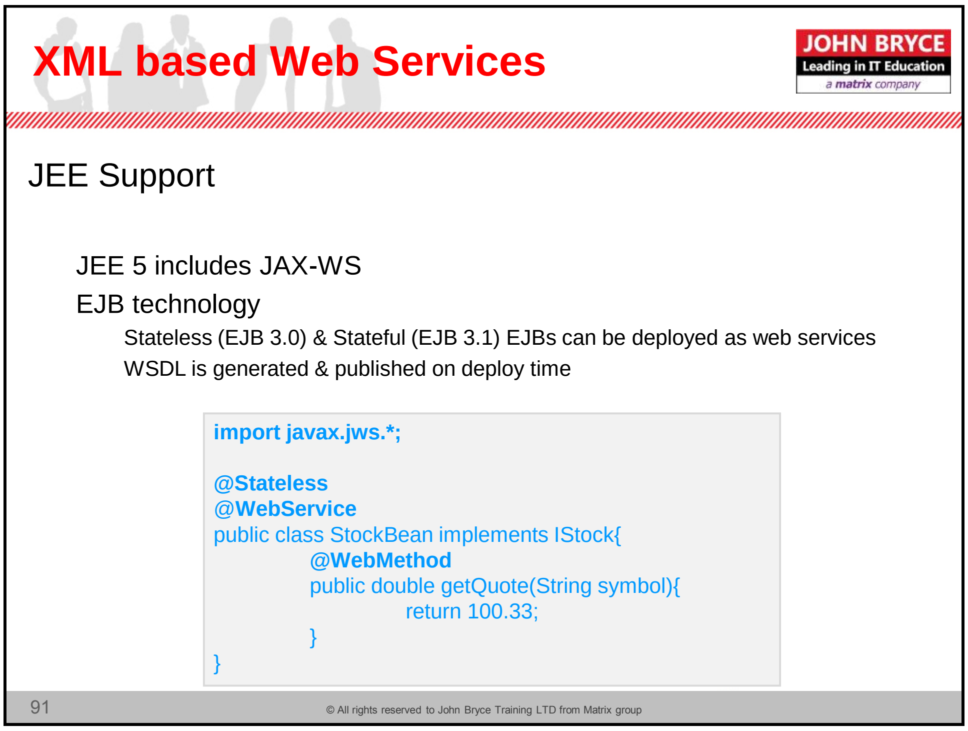
```
public class Client {
    public static void main(String[] args){

        BusinessService service = new BusinessService();
        Business calc = service.getBusinessPort();
        System.out.println("Call Started…");
        System.out.println(calc.sum(100,200));
        System.out.println("Call Ended…");
    }
}
```

Exercise

Lab 1 – Phase 2

In this exercise you are about create a standalone Java client to test your service





XML based Web Services

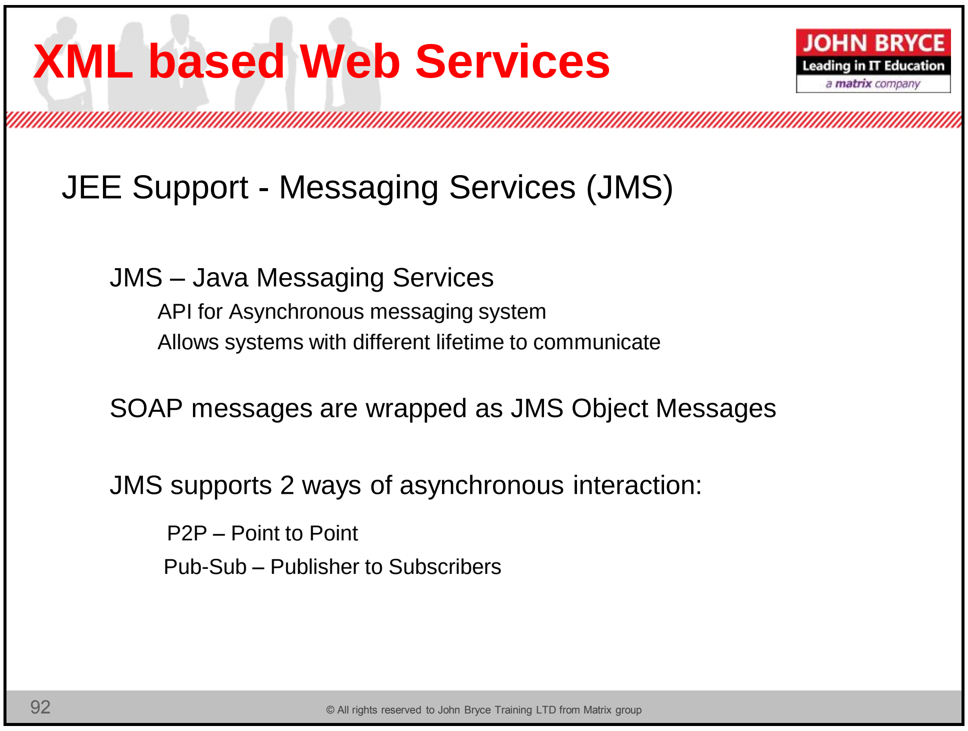
JOHN BRYCE Leading in IT Education a matrix company

JEE Support

- JEE 5 includes JAX-WS
- EJB technology
  - Stateless (EJB 3.0) & Stateful (EJB 3.1) EJBs can be deployed as web services
  - WSDL is generated & published on deploy time

```
import javax.jws.*;

@Stateless
@WebService
public class StockBean implements IStock{
		@WebMethod
		public double getQuote(String symbol){
				return 100.33;
		}
}
```

91



XML based Web Services

JOHN BRYCE Leading in IT Education a matrix company

JEE Support - Messaging Services (JMS)

- JMS – Java Messaging Services
  - API for Asynchronous messaging system
  - Allows systems with different lifetime to communicate
- SOAP messages are wrapped as JMS Object Messages
- JMS supports 2 ways of asynchronous interaction:
  - P2P – Point to Point
  - Pub-Sub – Publisher to Subscribers

92

REST based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

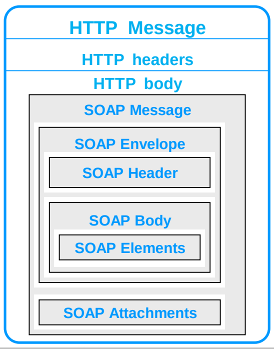
- Big data – the challenge
  - Parallel computing
  - NoSQL DBs
  - Saving bandwidth & faster response
- Introduction to REST
- HTTP for RPC
- JAX-RS - RESTful
  - Creating a Java service
  - Publishing & testing
  - Using *Jersey* client API for generating clients
  - Tokens & session management
  - WADL
  - JEE support

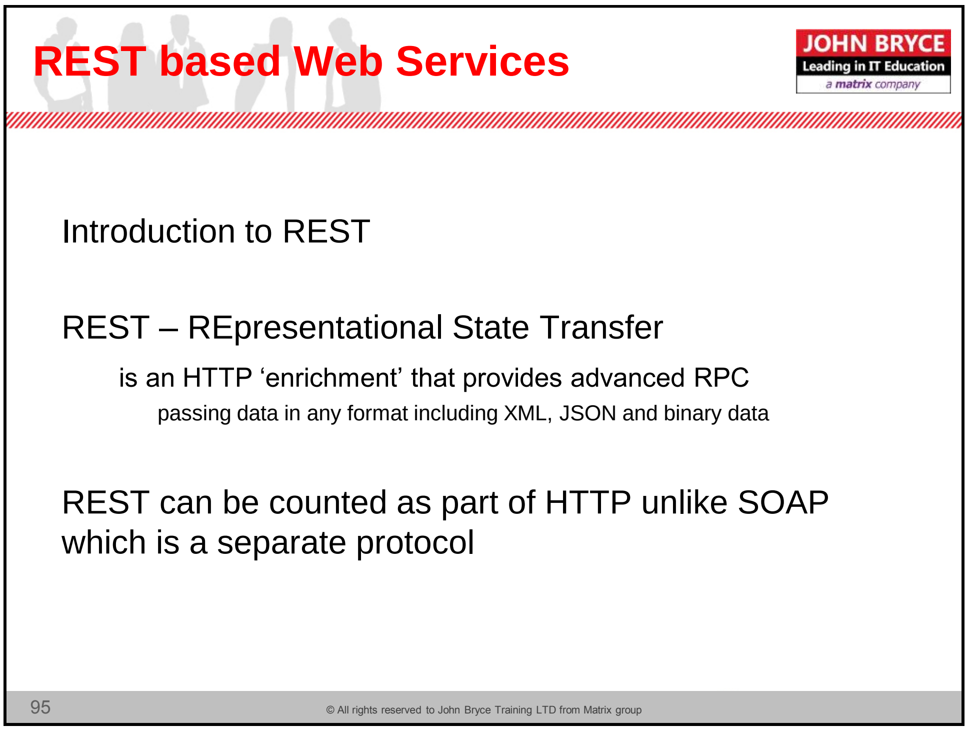
REST based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

Big data – the challenge

- From single & concurrent to parallel & cloud computing
- Using NoSQL DB in addition to the classic relational DB
- Saving bandwidth & performance
  - XML is a very inefficient protocol uses tags to wrap data
  - XML forces the use of parsers
  - SOAP is an additional protocol on top of HTTP





REST based Web Services

JOHN BRYCE Leading in IT Education a matrix company

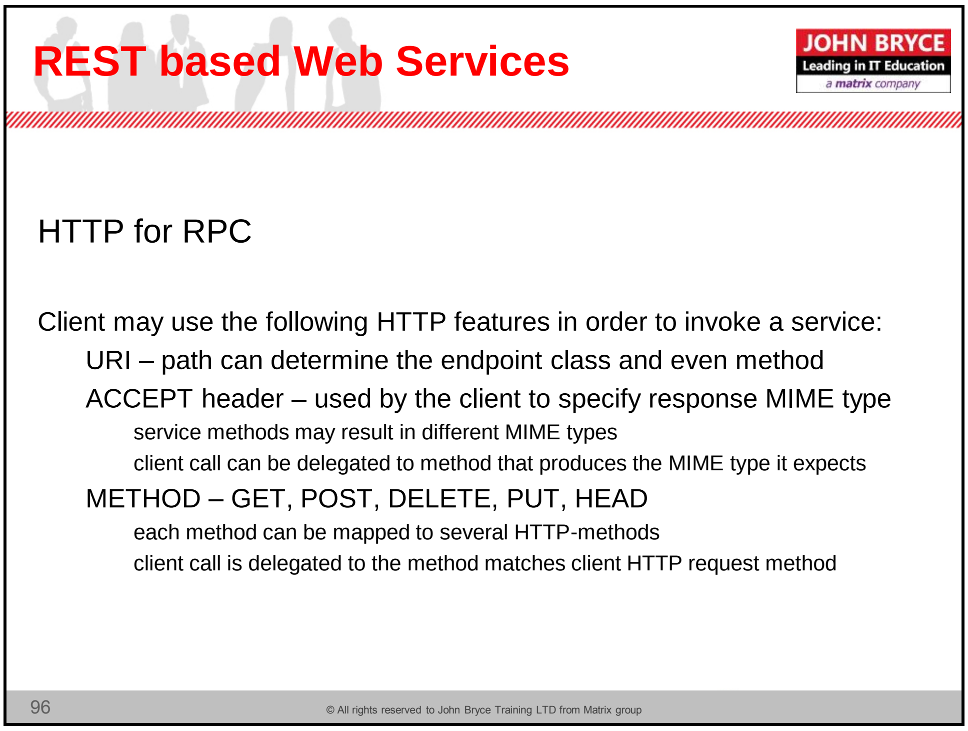
Introduction to REST

REST – REpresentational State Transfer

- is an HTTP ‘enrichment’ that provides advanced RPC
  - passing data in any format including XML, JSON and binary data

REST can be counted as part of HTTP unlike SOAP which is a separate protocol

95



REST based Web Services

JOHN BRYCE Leading in IT Education a matrix company

HTTP for RPC

Client may use the following HTTP features in order to invoke a service:

- URI – path can determine the endpoint class and even method
- ACCEPT header – used by the client to specify response MIME type
  - service methods may result in different MIME types
  - client call can be delegated to method that produces the MIME type it expects
- METHOD – GET, POST, DELETE, PUT, HEAD
  - each method can be mapped to several HTTP-methods
  - client call is delegated to the method matches client HTTP request method

96

REST based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

HTTP for RPC

Suggested way to implement business according to HTTP method

HTTP Method	Single element	Collection
GET	Fetch an element from a collection	Fetch the whole collection
PUT	Replace or create new element in a collection	Override one collection with a new one
POST	Assign a value to an object	Add new value to a collection
DELETE	Delete a specific element from a collection	Delete the entire collection

97

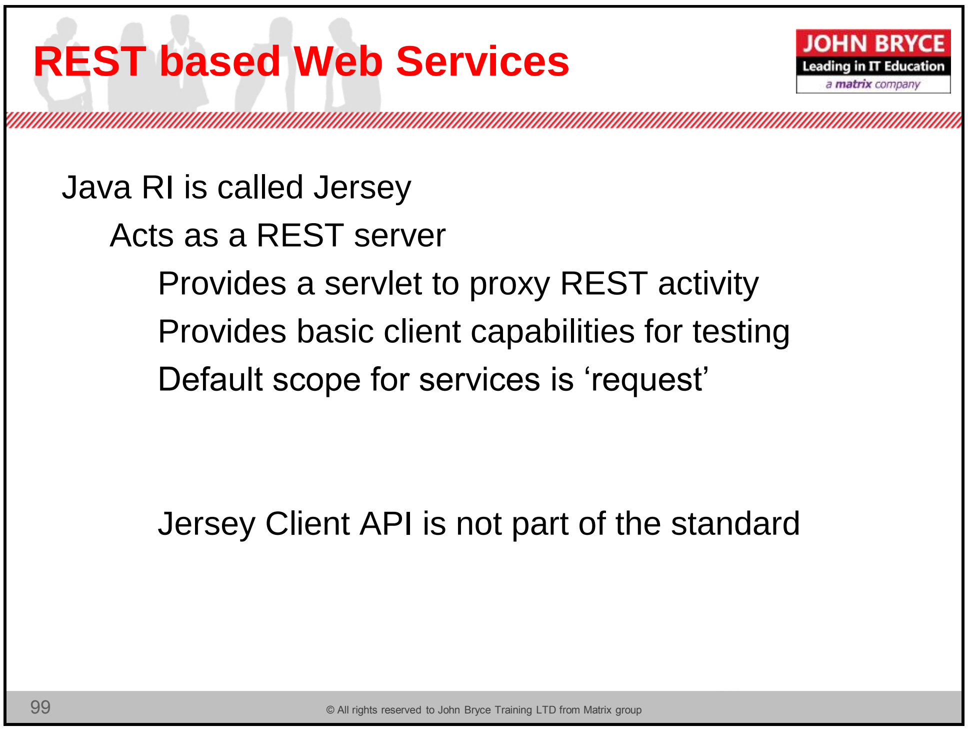
REST based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

JAX – RS - RESTful

- Java API for creating RESTful based web-services
  - Uses Jersey implementation as RI
- Uses annotations much like JAX-WS
- 4 principles to make it fast and simple:
  - Identify – tracking endpoints is based on URI
  - Unified interface – using HTTP methods (GET,POST,DELETE,PUT…)
  - Self descriptive content – XML , JSON…
  - Stateful interaction – by attaching session data or using tokens
- JAXB used for XML and JSON

98



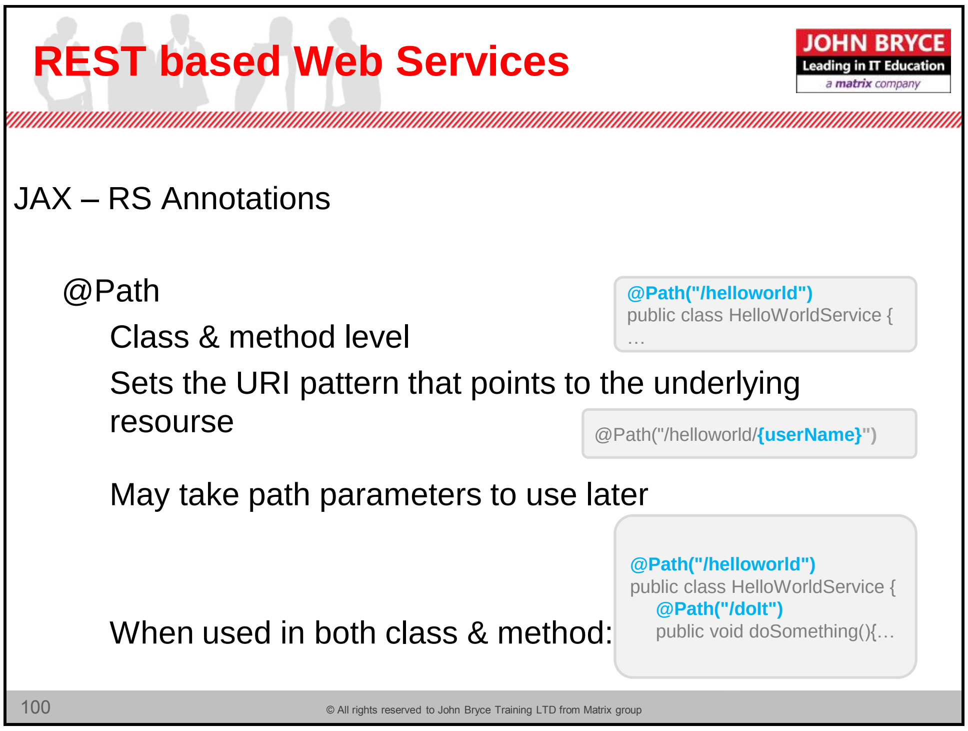
REST based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

- Java RI is called Jersey
  - Acts as a REST server
    - Provides a servlet to proxy REST activity
    - Provides basic client capabilities for testing
    - Default scope for services is ‘request’

    - Jersey Client API is not part of the standard

99



REST based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

JAX – RS Annotations

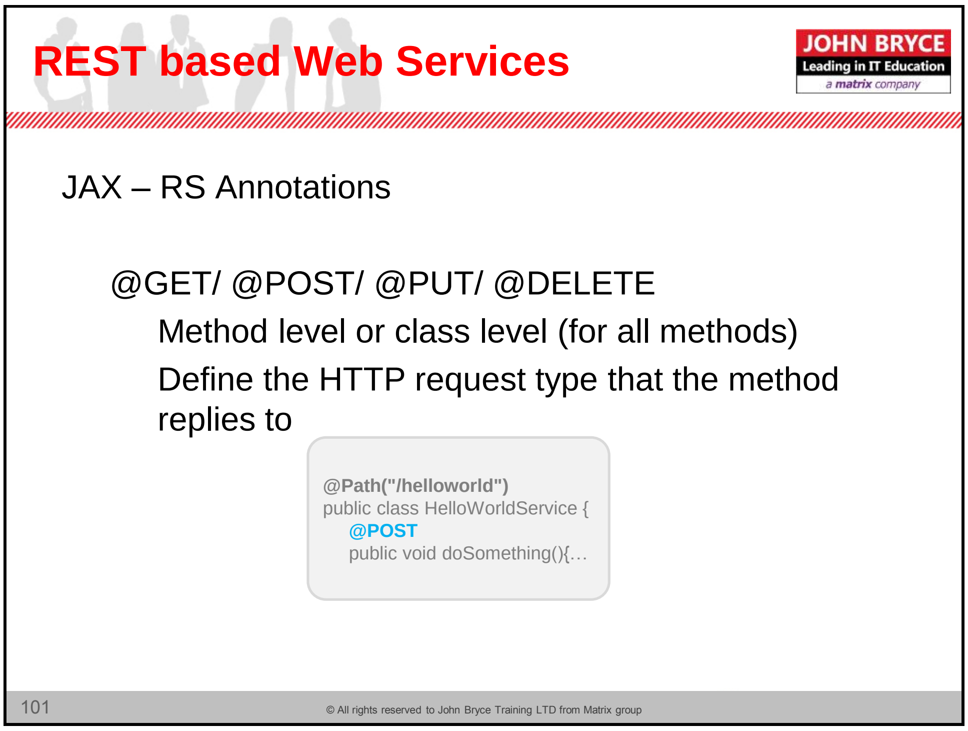
- @Path
  - Class & method level
  - Sets the URI pattern that points to the underlying resourse
  - May take path parameters to use later
  - When used in both class & method:

```
@Path("/helloworld")
public class HelloWorldService {
…
```

```
@Path("/helloworld/{userName}")
```

```
@Path("/helloworld")
public class HelloWorldService {
    @Path("/doIt")
    public void doSomething(){…
```

100



REST based Web Services

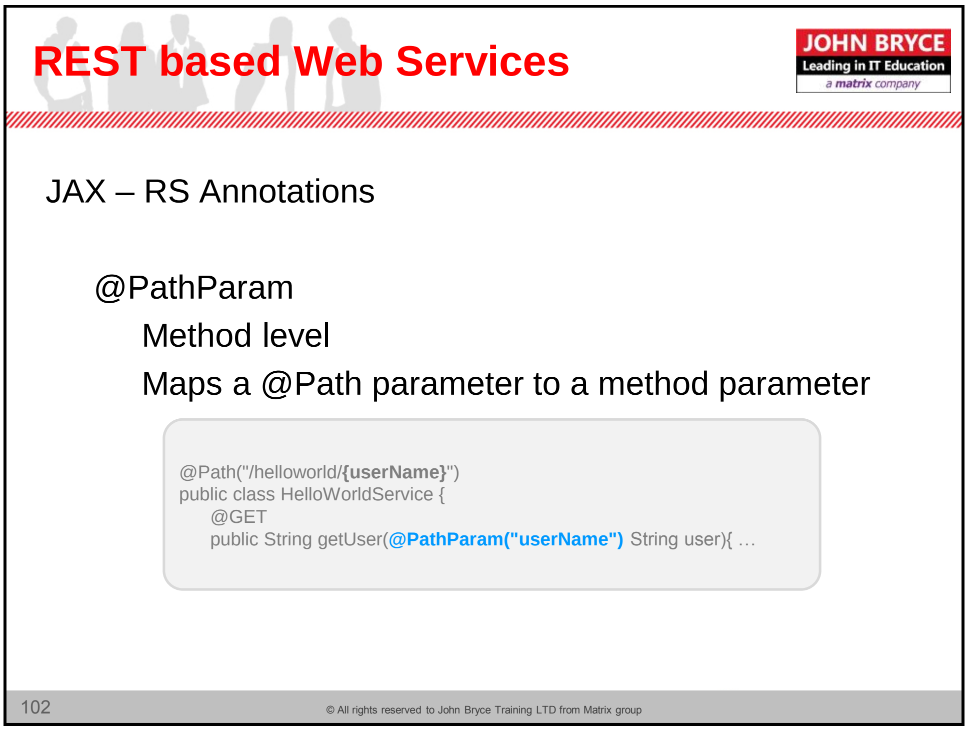
JOHN BRYCE
Leading in IT Education
a matrix company

JAX – RS Annotations

- @GET/ @POST/ @PUT/ @DELETE
  - Method level or class level (for all methods)
  - Define the HTTP request type that the method replies to

```
@Path("/helloworld")
public class HelloWorldService {
    @POST
    public void doSomething(){…
```

101 © All rights reserved to John Bryce Training LTD from Matrix group



REST based Web Services

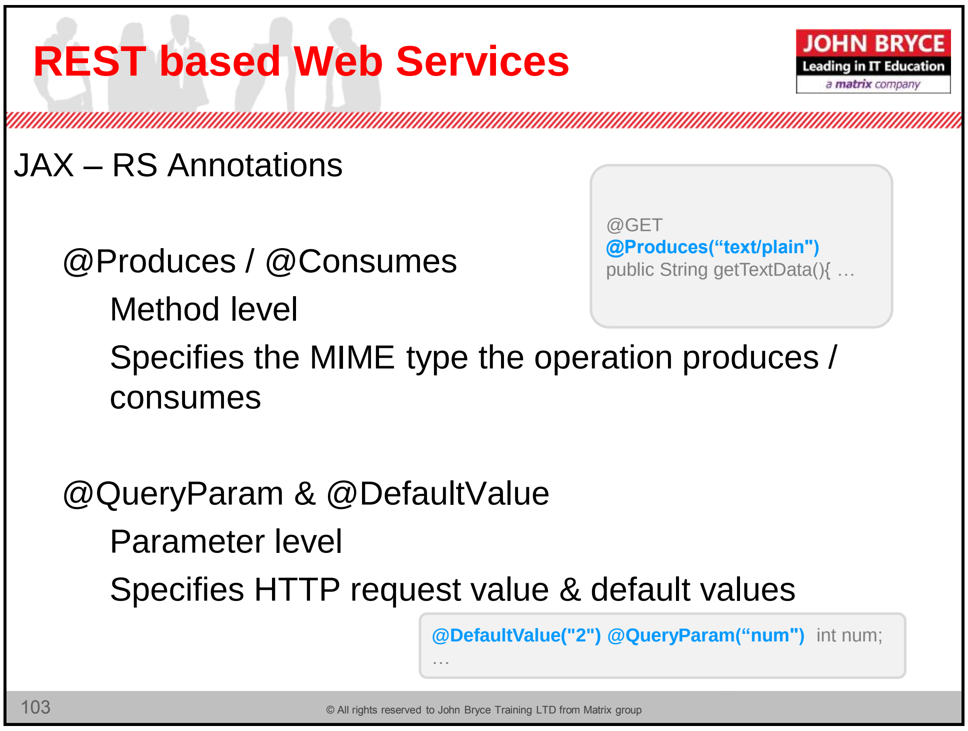
JOHN BRYCE
Leading in IT Education
a matrix company

JAX – RS Annotations

- @PathParam
  - Method level
  - Maps a @Path parameter to a method parameter

```
@Path("/helloworld/{userName}")
public class HelloWorldService {
    @GET
    public String getUser(@PathParam("userName") String user){ …
```

102 © All rights reserved to John Bryce Training LTD from Matrix group



REST based Web Services

JOHN BRYCE Leading in IT Education a matrix company

JAX – RS Annotations

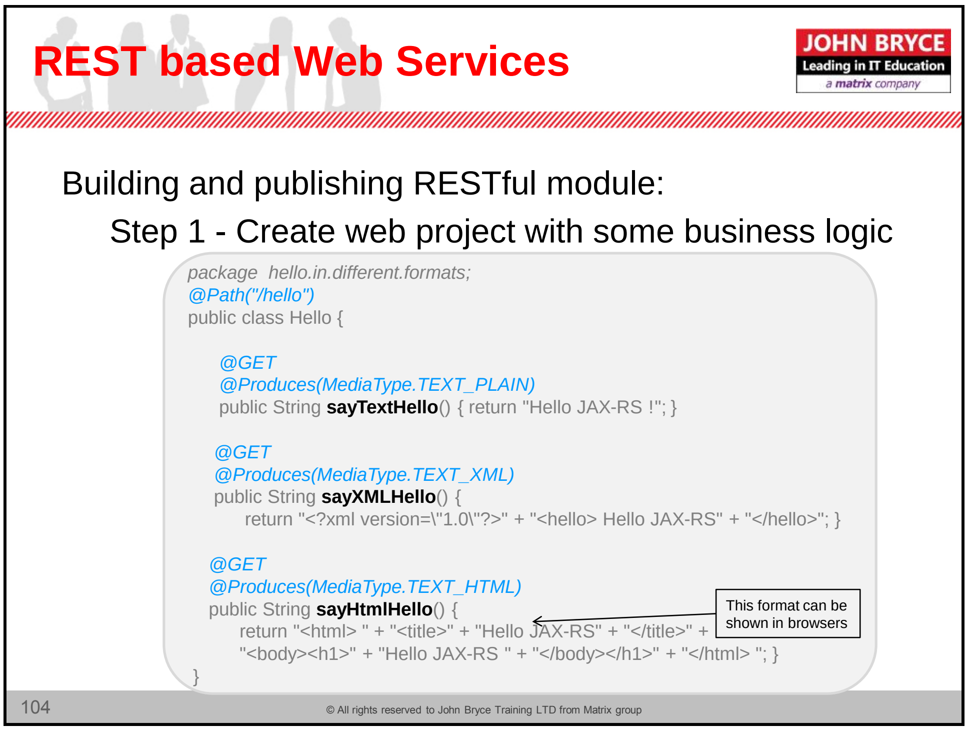
- @Produces / @Consumes
  - Method level
  - Specifies the MIME type the operation produces / consumes

```
@GET
@Produces(“text/plain")
public String getTextData(){ …
```

- @QueryParam & @DefaultValue
  - Parameter level
  - Specifies HTTP request value & default values

```
@DefaultValue("2") @QueryParam(“num")  int num;
…
```

103



REST based Web Services

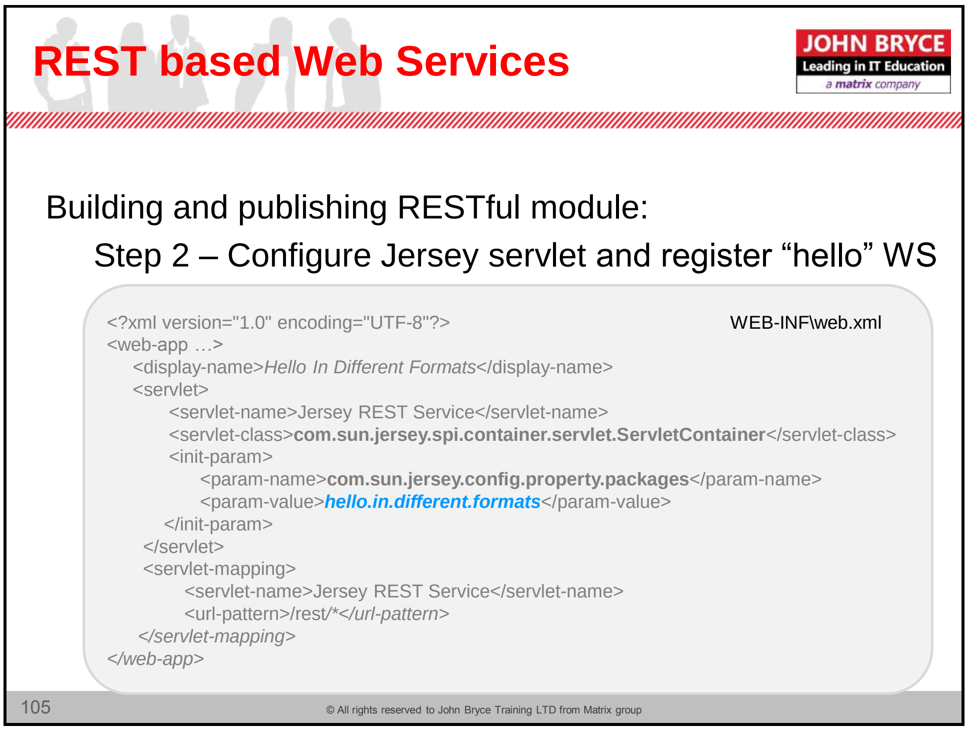
JOHN BRYCE Leading in IT Education a matrix company

- Building and publishing RESTful module:
  - Step 1 - Create web project with some business logic

```
package  hello.in.different.formats;
@Path("/hello")
public class Hello {

     @GET
     @Produces(MediaType.TEXT_PLAIN)
     public String **sayTextHello**() { return "Hello JAX-RS !"; }

    @GET
    @Produces(MediaType.TEXT_XML)
    public String **sayXMLHello**() {
         return "<?xml version=\"1.0\"?>" + "<hello> Hello JAX-RS" + "</hello>"; }

   @GET
   @Produces(MediaType.TEXT_HTML)
   public String **sayHtmlHello**() {
        return "<html> " + "<title>" + "Hello JAX-RS" + "</title>" +
        "<body><h1>" + "Hello JAX-RS " + "</body></h1>" + "</html> "; }
}
```

This format can be shown in browsers

104



REST based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

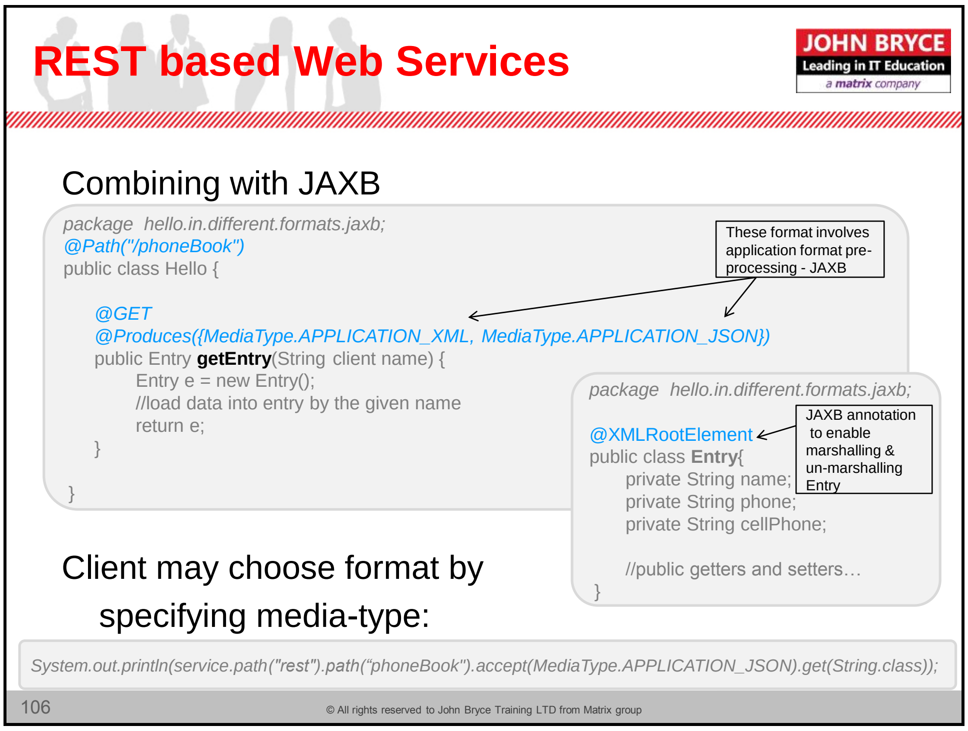
Building and publishing RESTful module:

Step 2 – Configure Jersey servlet and register “hello” WS

WEB-INF\web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app …>
   <display-name>Hello In Different Formats</display-name>
   <servlet>
        <servlet-name>Jersey REST Service</servlet-name>
        <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
        <init-param>
            <param-name>com.sun.jersey.config.property.packages</param-name>
            <param-value>hello.in.different.formats</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
          <servlet-name>Jersey REST Service</servlet-name>
          <url-pattern>/rest/*</url-pattern>
    </servlet-mapping>
</web-app>
```

105 © All rights reserved to John Bryce Training LTD from Matrix group



REST based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

Combining with JAXB

```
package  hello.in.different.formats.jaxb;
@Path("/phoneBook")
public class Hello {

    @GET
    @Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
    public Entry getEntry(String client name) {
          Entry e = new Entry();
          //load data into entry by the given name
          return e;
    }
}
```

These format involves application format pre-processing - JAXB

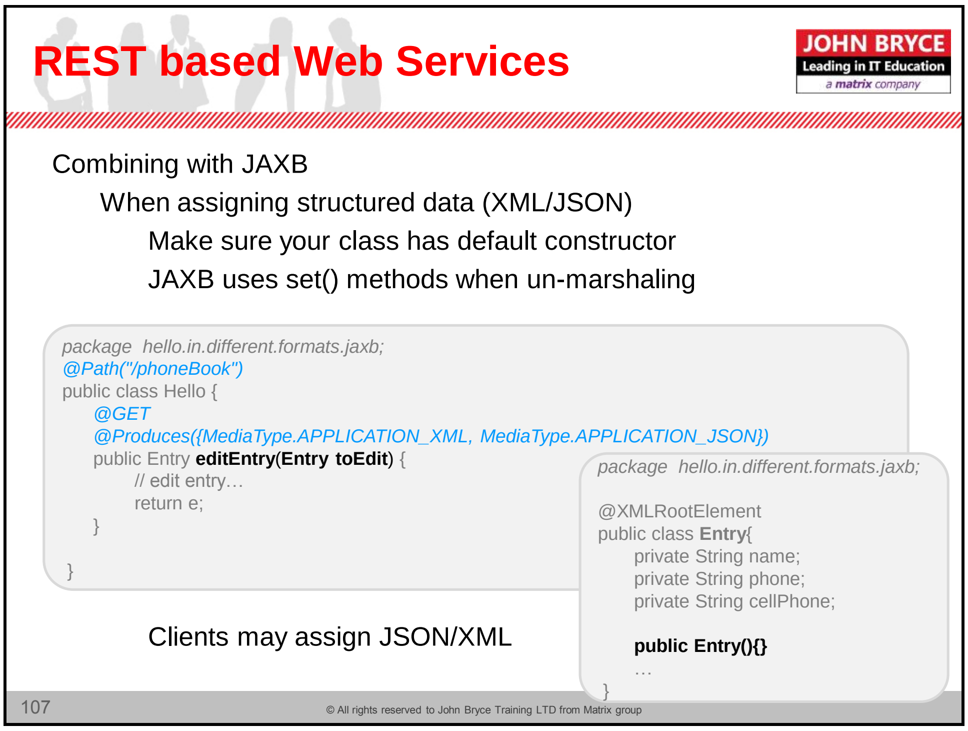
```
package  hello.in.different.formats.jaxb;

@XMLRootElement
public class Entry{
     private String name;
     private String phone;
     private String cellPhone;

     //public getters and setters…
}
```

JAXB annotation to enable marshalling & un-marshalling Entry

Client may choose format by specifying media-type:

```
System.out.println(service.path("rest").path(“phoneBook").accept(MediaType.APPLICATION_JSON).get(String.class));
```

106 © All rights reserved to John Bryce Training LTD from Matrix group



REST based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

Combining with JAXB

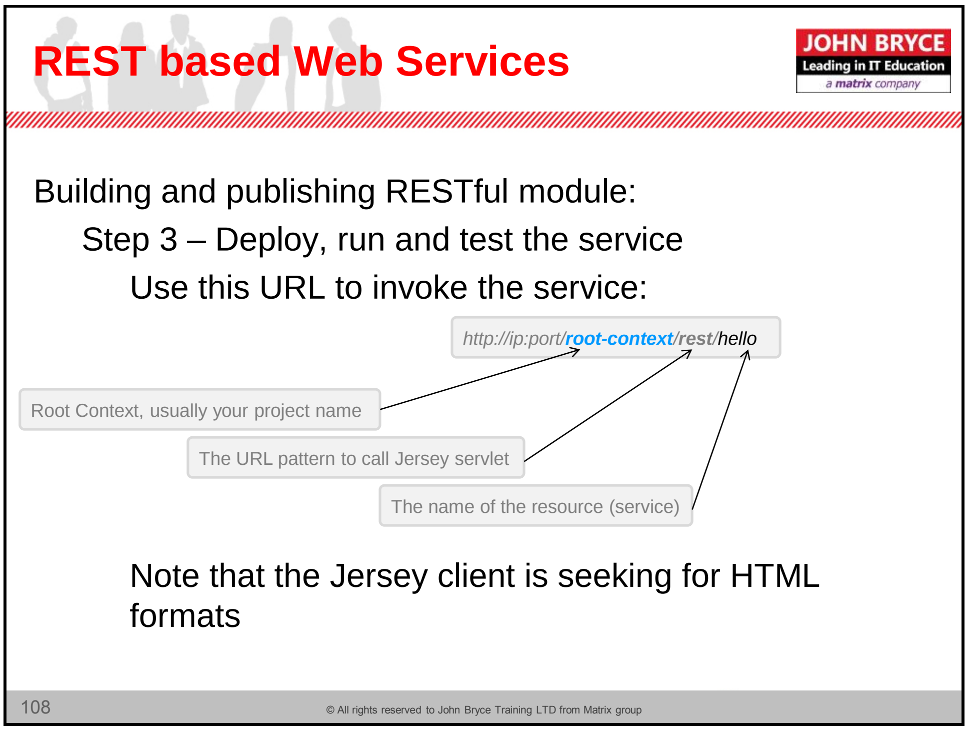
- When assigning structured data (XML/JSON)
  - Make sure your class has default constructor
  - JAXB uses set() methods when un-marshaling

```
package  hello.in.different.formats.jaxb;
@Path("/phoneBook")
public class Hello {
     @GET
     @Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
     public Entry editEntry(Entry toEdit) {
            // edit entry…
            return e;
     }

}
```

```
package  hello.in.different.formats.jaxb;

@XMLRootElement
public class Entry{
      private String name;
      private String phone;
      private String cellPhone;

      public Entry(){}
      …
}
```

- Clients may assign JSON/XML

107 © All rights reserved to John Bryce Training LTD from Matrix group



REST based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

Building and publishing RESTful module:

- Step 3 – Deploy, run and test the service
  - Use this URL to invoke the service:

http://ip:port/**root-context**/rest/hello

Root Context, usually your project name

The URL pattern to call Jersey servlet

The name of the resource (service)

- Note that the Jersey client is seeking for HTML formats

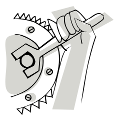
108 © All rights reserved to John Bryce Training LTD from Matrix group

Exercise

JOHN BRYCE
Leading in IT Education
a matrix company

Lab 2 – Phase 1

In this exercise you are about create and publish Java POJO as REST based Web service using JAX-RS



REST based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

Creating RESTful client

Jersey offers simple client API

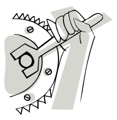
```
public class Test {
   public static void main(String[] args) {
       ClientConfig config = new DefaultClientConfig();
       Client client = Client.create(config);
       URI baseUri = UriBuilder.fromUri("http://ip:port/hello.in.different.formats").build();
       WebResource service = client.resource(baseUri );
       // Get plain text
      System.out.println(service.path("rest").path("hello").accept(MediaType.TEXT_PLAIN).get(String.class));
      // Get XML
      System.out.println(service.path("rest").path("hello").accept(MediaType.TEXT_XML).get(String.class));
     // The HTML
     System.out.println(service.path("rest").path("hello").accept(MediaType.TEXT_HTML).get(String.class));
}
```

get(..) / post(..)
/ put(..) / delete(..)

Exercise

Lab 2 – Phase 2

In this exercise you are about create a standalone Java client to test your service

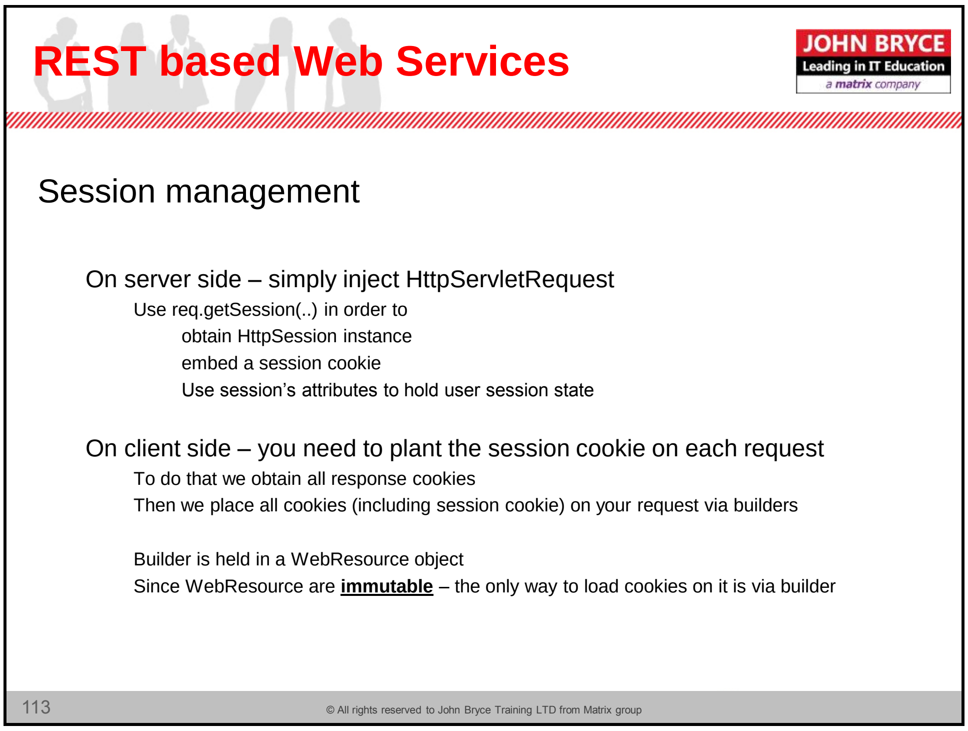


REST based Web Services

Working with CGI

- It is very easy to use Servlets API in your services
- Why would we do it?
  - Scope management request, session, application
  - Do some custom request / response processing
- How do we use it?
  - Simply inject anything needed from Servlets API
    - Request, response, context…
    - Use @Context

```
@Path("/phoneBook")
public class Hello {

    @Context  private  HttpServletRequest  req;
    @Context  private  ServletContext  ctx;
    …
}
```



REST based Web Services

JOHN BRYCE Leading in IT Education a matrix company

Session management

On server side – simply inject HttpServletRequest

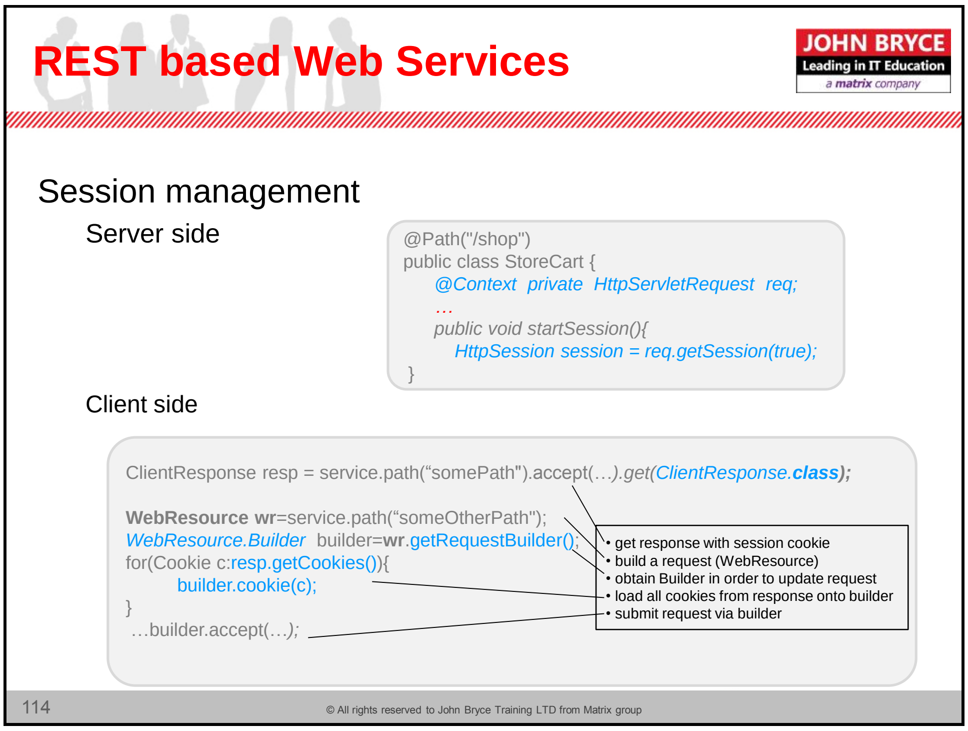
- Use req.getSession(..) in order to
  - obtain HttpSession instance
  - embed a session cookie
  - Use session’s attributes to hold user session state

On client side – you need to plant the session cookie on each request

- To do that we obtain all response cookies
- Then we place all cookies (including session cookie) on your request via builders

- Builder is held in a WebResource object
- Since WebResource are **immutable** – the only way to load cookies on it is via builder

113 © All rights reserved to John Bryce Training LTD from Matrix group



REST based Web Services

JOHN BRYCE Leading in IT Education a matrix company

Session management

Server side

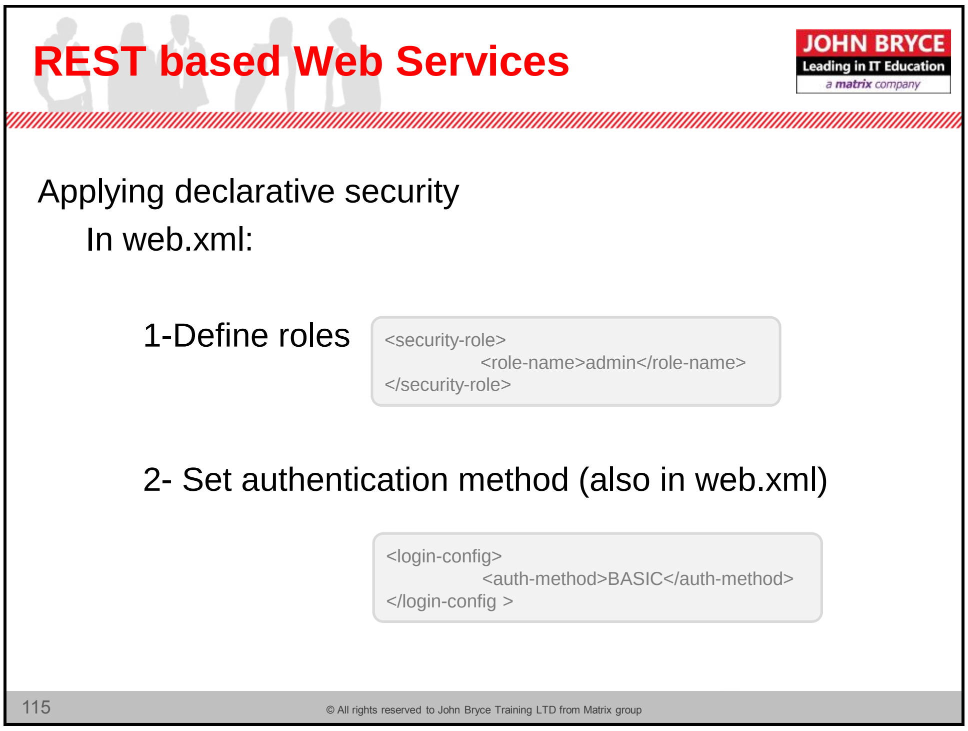
```
@Path("/shop")
public class StoreCart {
     @Context  private  HttpServletRequest  req;
     …
     public void startSession(){
         HttpSession session = req.getSession(true);
 }
```

Client side

```
ClientResponse resp = service.path(“somePath").accept(…).get(ClientResponse.class);

WebResource wr=service.path(“someOtherPath");
WebResource.Builder  builder=wr.getRequestBuilder();
for(Cookie c:resp.getCookies()){
          builder.cookie(c);
}
 …builder.accept(…);
```

- get response with session cookie
- build a request (WebResource)
- obtain Builder in order to update request
- load all cookies from response onto builder
- submit request via builder

114 © All rights reserved to John Bryce Training LTD from Matrix group



REST based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

Applying declarative security

In web.xml:

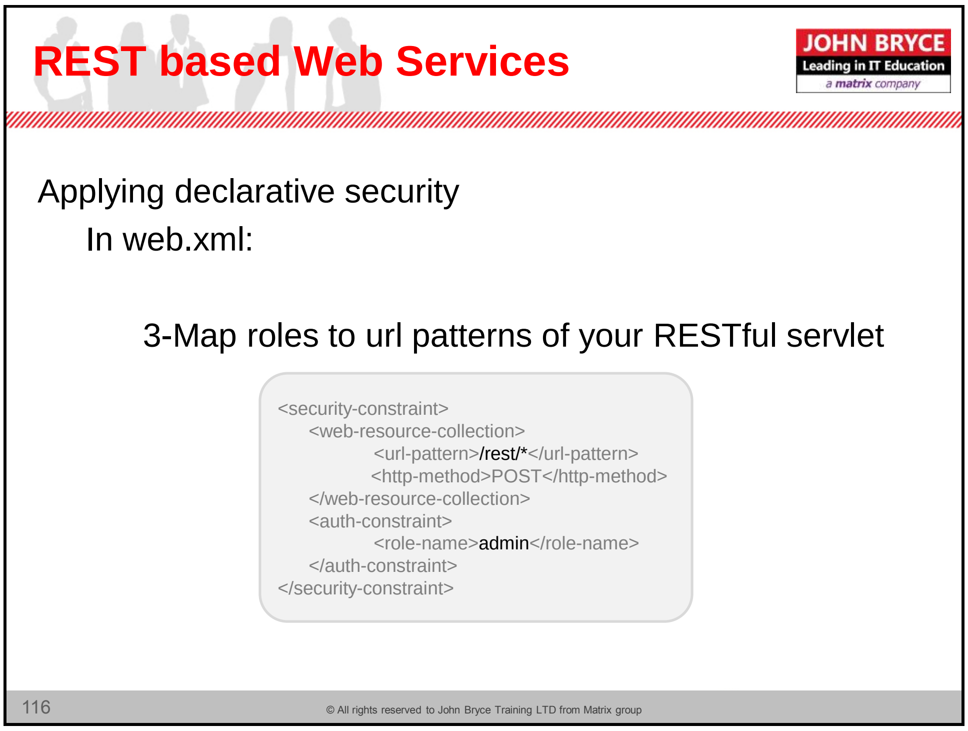
1-Define roles

```
<security-role>
                <role-name>admin</role-name>
</security-role>
```

2- Set authentication method (also in web.xml)

```
<login-config>
                <auth-method>BASIC</auth-method>
</login-config >
```

115 © All rights reserved to John Bryce Training LTD from Matrix group



REST based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

Applying declarative security

In web.xml:

3-Map roles to url patterns of your RESTful servlet

```
<security-constraint>
     <web-resource-collection>
                <url-pattern>/rest/*</url-pattern>
               <http-method>POST</http-method>
     </web-resource-collection>
     <auth-constraint>
                <role-name>admin</role-name>
     </auth-constraint>
</security-constraint>
```

116 © All rights reserved to John Bryce Training LTD from Matrix group

REST based Web Services

Applying declarative security

4- Use security annotations to declare roles access on your service

```
package  hello.in.different.formats;
@Path("/hello")
@RolesAllowed({“admin”,”guest”})
public class Hello {

    @RolesAllowed(“admin”)
    @GET
    @Produces(MediaType.TEXT_XML)
    public String sayXMLHello() {
        return "<?xml version=\"1.0\"?>" + "<hello> Hello JAX-RS" + "</hello>"; }

    @PermitAll
    @GET
    @Produces(MediaType.TEXT_HTML)
    public String sayHtmlHello() {
        return "<html> " + "<title>" + "Hello JAX-RS" + "</title>" +
        "<body><h1>" + "Hello JAX-RS " + "</body></h1>" + "</html> "; }
}
```

Security annotations:
- @RolesAllowed – lists permitted roles
- @DenyAll – allows non-logged users
- @PermitAll – permits all declared roles
- None – available to anyone

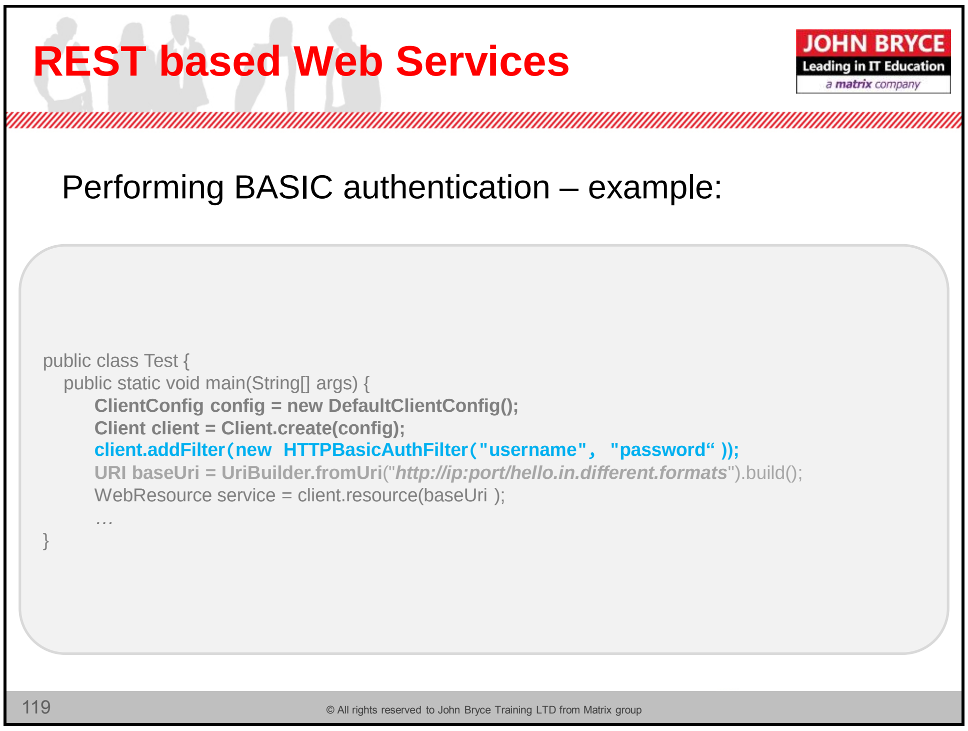
REST based Web Services

Performing BASIC authentication

Add ‘HTTP Basic Authentication’ header to the HTTP

May use Jersey HttpBasicAuthFilter to do that

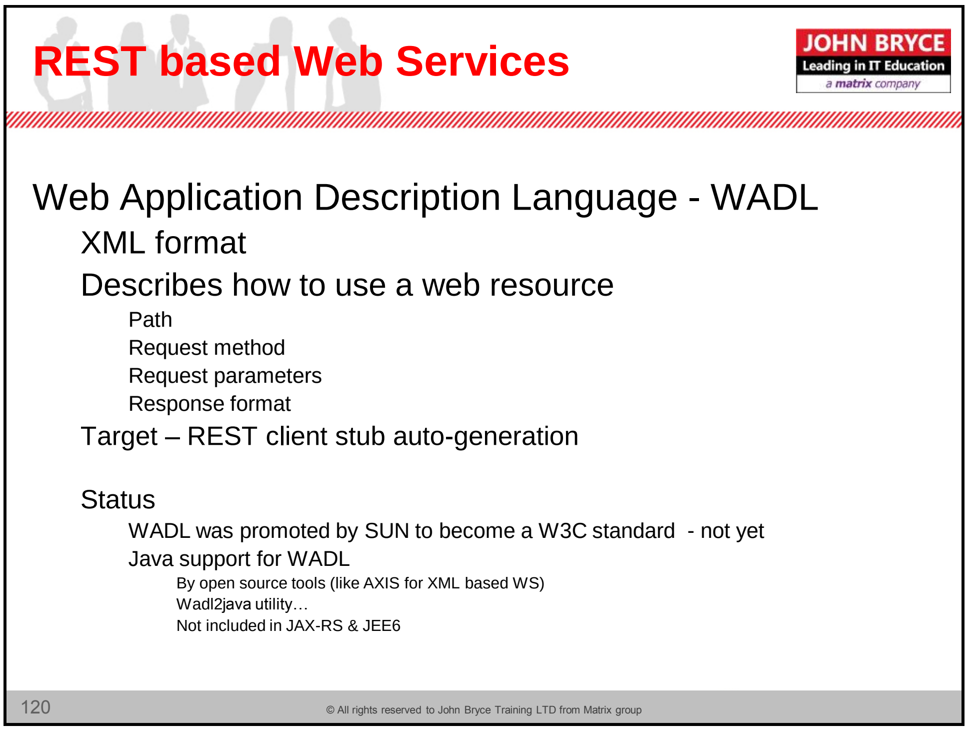
- Adds the header only if doesn’t exist
- Accepts username and password
- Username and password are for creating user Principal
- Principals are then mapped to application roles



REST based Web Services

Performing BASIC authentication – example:

```
public class Test {
   public static void main(String[] args) {
        ClientConfig config = new DefaultClientConfig();
        Client client = Client.create(config);
        client.addFilter(new  HTTPBasicAuthFilter("username",  "password“ ));
        URI baseUri = UriBuilder.fromUri("http://ip:port/hello.in.different.formats").build();
        WebResource service = client.resource(baseUri );
        …
}
```



REST based Web Services

Web Application Description Language - WADL

- XML format
- Describes how to use a web resource
  - Path
  - Request method
  - Request parameters
  - Response format
- Target – REST client stub auto-generation

- Status
  - WADL was promoted by SUN to become a W3C standard - not yet
  - Java support for WADL
    - By open source tools (like AXIS for XML based WS)
    - Wadl2java utility…
    - Not included in JAX-RS & JEE6

REST based Web Services

WADL

Main elements

- <application> - the root element
- <grammars> - includes *.xsd which may describe response content (if it is XML based) – optional
- <resources> - contains all the resources described in the document
- <resource> - describe the resource itself, holds the path to it
  - <method> - describes a method for the invocation (GET/POST….)
    - <request> - describes the request and response structure
      - <param> <option> - describe a parameter name, type (xsd:) and optional values
      - <representation> - specifies request body MIME-TYPE
    - <response> - describes the response status code and its MIME type
      - <representation> - specifies response body MIME-TYPE

REST based Web Services

WADL – simple example

```
<application xmlns="http://wadl.dev.java.net/2009/02">
        <resources base="http://example.co.il/rest">
                <resource path="employees">
                                <method name="GET"/>
                                <method name="POST"/>
                </resource>
        </resources>
 </application>
```

Calling to http://example.co.il/rest/employees
is available via POST & GET
POST – probably add a new employee
GET – fetch employee list

REST based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

WADL - placeholder for path-param

```
<application xmlns="http://wadl.dev.java.net/2009/02">
     <resources base="http://example.co.il/rest">
          <resource path="employees">
                   <resource path="{empId}">
                            <param  required="true" name="empID"/>
                            <method name="GET"/>
                   </resource>
          </resource>
     </resources>
</application>
```

Calling to http://example.co.il/rest/employees/1234
is available via GET
empId will get the value 1234 and delegates it to the empID method parameter

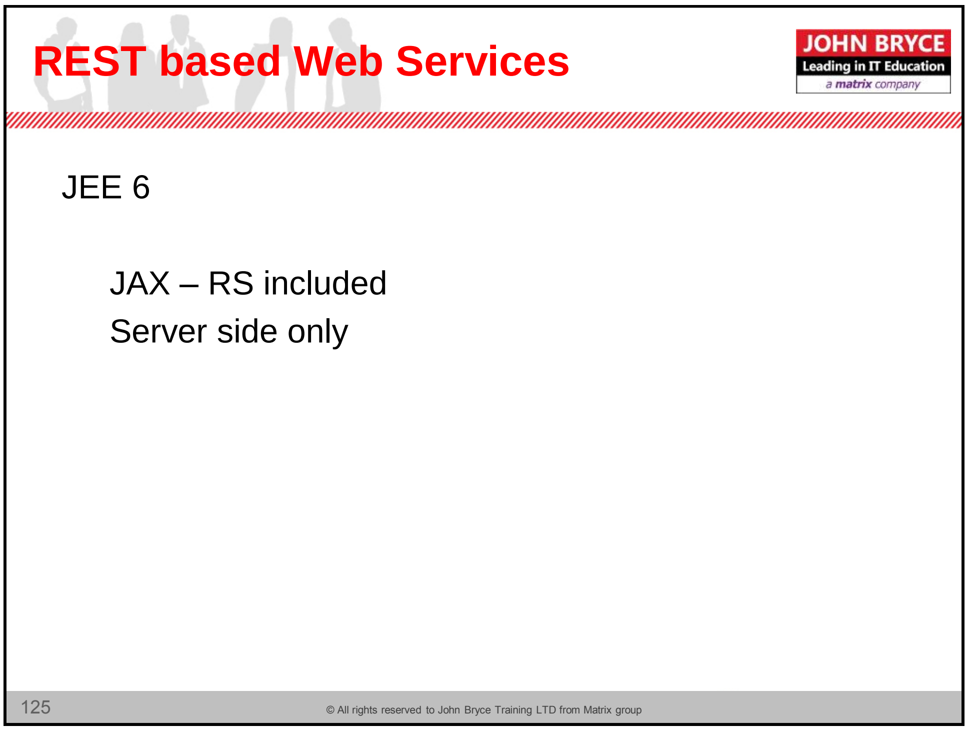
REST based Web Services

JOHN BRYCE
Leading in IT Education
a matrix company

WADL - placeholder for query-param

```
<application xmlns="http://wadl.dev.java.net/2009/02">
     <resources base="http://example.co.il/rest">
          <resource path="employees">
              <method name="GET">
                   <request>
                            <param  required="false" default="1" name="empID"/>
                     </request>
             </method>
          </resource>
     </resources>
</application>
```

Calling to http://example.co.il/rest/employees?1234
is available via GET
empId will get the value 1234 and delegates it to the empID method parameter



REST based Web Services

JEE 6

- JAX – RS included
- Server side only

Java Web Services Lab Guide

Written by Rony Keren
Internet Team
John Bryce Training Center

Version: 1.0

Lab 1 - JAX-WS

Item Stock Web Service

Phase 1 – Server Side

- Create basic Java project
- Create class store.Item
  - attributes:
    - name : String
    - category : String
    - price : float
    - amount : int
  - methods
    - getters / setters
    - override toString() to print item details
    - override equals(Object o) to check according to name, price & category
    - Add default constructor
    - Add another constructor that takes name, category & price
- Create service class store.StoreService
  - attribute:
    - stock : ArrayList<Item>
  - methods:
    - addItem(Item) : void
    - removeItem(Item) : void
    - getItem(String name, String category) : Item
    - getItemsByCategory(String category) : Item[]
    - getAllItems(String category) : Item[]
    - getAmount() : int
  - Map class & methods to become a web-service via JAX-WS annotations
  - Use DOCUMENTED style
    - So Item schema (XSD) is generated
    - Client will use stand alone classes rather than inner classes
- Create PublishStoreWS class with main method
  - Publish your service to http://localhost:8080/StoreWS/store
  - Test the generated WSDL & Item schema

Phase 2 – Client Side

- While your service is running, use wsimport utility to generate client stub and types
  - Make sure that the destination directory exists before using wsimport
- Create a new Java project
- Drag the generated files to your project and place in the matching package
- Create a TestStoreWS client to verify all works fine

Lab 2 - JAX-RS

Item Stock Web RESTful Service

Phase 1 – Server Side

- Create dynamic web project
- Drag all the required jars to Web Content\ WEB-INF\lib
  - you'll find these file in the exercise directory : exercise\WEB-INF\lib
- Drag web.xml to Web Content\ WEB-INF
  - you'll find these file in the exercise directory : exercise\WEB-INF
- Import the 2 provided classes for your service (found at exercise\src):
  - items.Item
    - attributes:
      - name : String
      - price : float
    - methods
      - getters / setters
      - toString() to print item details
      - equals(Object o) to check according to name& price
      - default constructor
      - another constructor that takes name & price
    - NOTE: it is denoted with JAXB annotation - @XMLRootElement so items can be marshaled & un-marshaled
  - items.ItemCart
    - attribute:
      - cart : ArrayList<Item>
    - methods:
      - addItem(Item) : void
      - removeItem(String itemName) : void
      - getAll() : List<Item>
      - getItems (String itemName) : Item
- Create items.ItemService class to enable REST based interaction:
  - Session management is done via tokens
    - Define an attribute Map<String,ItemCart>
    - Each entry is a 'client session'. String is used as a session token
  - Add the following methods
    - createCart()
      - This method randomize a token (number) for the client, instantiates an empty ItemCart and puts both in the map collection for future use.
      - invoked via GET & uses a dedicated path
      - returns the token as plain text so it can be used as a path parameter on the next calls.

- getAllItems(String token) : Item[]
  - Fetches all items for the client according to a given token
  - invoked via GET
  - returns array of items as APPLICATION_JSON
- addItem(String token, String name, String price) : String
  - creates a new Item and adds it to the client cart
  - invoked via POST
  - returns itemName+" added" as plain text
- removeItem(String token, String name) : String
  - removes the specified item from client cart
  - invoked via DELETE
  - itemName+"removed" as plain text

- Edit web-xml
  - register Jersey servlet & initialize it with the package of your new service
  - map the servlet to a url pattern – like "rest\jaxb"
- Deploy the project to TomCat web server and launch server
- Make sure there is no deployment errors & take the server down

Phase 2 – Client Side

- In the same project (where all Jersey implementations are set already) create a new package – 'client'
- Create a client.RESTfullItemClient class with main method
- Implement main to test your service:
  - first build the resource and connect
  - than call 'createCart' to obtain a token
  - now, use the token to invoke addItem, removeItem, getItem & getAllItems to verify that session state is maintained and that the service works well.
- In order to run your client:
  - first, run the whole project on server (client main is ignored)
  - run your test as a Java application